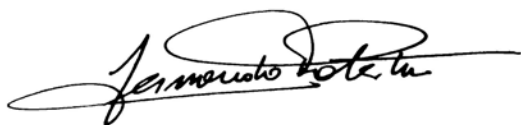
VOI E IL VOSTRO COMMODORE 64

Fulvio Francesconi
Fernando Paterlini

VOI E IL VOSTRO COMMODORE 64

Fulvio Francesconi
Fernando Paterlini

GRUPPO
EDITORIALE
JACKSON
Via Rosellini, 12
20124 Milano

E' doveroso per noi citare alcune persone senza le quali QUESTO non sarebbe mai stato possibile. In particolar modo ringraziamo:

LUISA CINZIA RAFFAELLA

GIANMARIA ROGER BRANCO

MARA ed OTTONE

e tutti Coloro che direttamente o indirettamente ci hanno aiutato, compresi i componenti della Redazione del Gruppo Editoriale Jackson che ci hanno accolto nella loro grande FAMIGLIA.

Io ringrazio invece questo libro perche' mi ha permesso di ritrovare un vecchio amico e di passare con lui molte, sottolineo molte, ore di piacevole compagnia, anche se sotto il torchio del dovere, rendendomele liete.

Il Gruppo Editoriale Jackson ringrazia per il prezioso lavoro svolto nella stesura dell'edizione originale la signora Francesca Di Fiore e l'ing. Roberto Pancaldi.

Fotocomposizione: Lineacomp S.r.l. - Via Rosellini, 12 - 20124 Milano

Stampato in Italia da:
S.p.A. Alberto Matarelli - Milano - Stabilimento Grafico

alle nostre mogli ed alla pazienza
che hanno saputo o dovuto dimostrare

INDICE

CAPITOLO 6

CAPITOLO 7

CAPITOLO 8

CAPITOLO 9

CAPITOLO 10

LEGGETEMI!

Io non mi chiamo 'introduzione' proprio perche' voglio che voi mi leggiate.

Mi spiego: normalmente la maggior parte delle persone quando apre un libro salta la parte introduttiva. E' molto importante che voi mi leggiate perche' vi do' delle indicazioni utili sulle cose che sono contenute in questo libro, sugli scopi che questo libro si propone di raggiungere e su come dovete leggerlo.

Leggendo questo libro voi potrete:
1) imparare ad operare correttamente col vostro COMMODORE 64
2) imparare i primi rudimenti del linguaggio basic applicato al COMMODORE 64.

Non intendiamo scrivere un trattato sul linguaggio basic: dipendera' dalle capacita' di ognuno, applicando correttamente le indicazioni che noi diamo sul linguaggio, sviluppare dette conoscenze per applicazioni di tipo professionale.

I programmi che noi Vi insegneremo a scrivere con questo computer potrebbero essere definiti esercitazioni comunque molto semplici, ma sono la base su cui appoggerete i vostri futuri sforzi nel campo della programmazione. Ognuno dei programmi che vi insegnamo a scrivere nel libro sono descritti riga per riga: ossia ogni riga contenente istruzioni viene descritta in modo che voi possiate capire esattamente che funzione ha quella riga nell'ambito di un programma in basic. Questo vi permettera' di utilizzare tali istruzioni nei programmi che voi stessi scriverete in futuro.
Ho parlato di uno degli scopi del libro. Per quanto riguarda l'altro scopo, ossia quello di insegnarvi ad operare correttamente col vostro COMMODORE 64, vi verranno date delle utili indicazioni su alcune norme fondamentali per aver cura della vostra apparecchiatura, dei supporti (vedi cassette magnetiche e simili), delle periferiche e sulle funzioni particolari del vostro COMMODORE 64.

Per quanto riguarda l'approfondimento dei singoli argomenti trattati nel libro vi verranno indicati di

volta in volta dei supporti a cui far riferimento.

BRAVI!!! Mi avete letto...

Ora pero' dovete osservare questa norma fondamentale: leggete tutte le parti del libro, ma non disordinatamente bensi' seguendo passo passo capitolo per capitolo quanto abbiamo scritto. Tutto e' collegato in modo logico. Quindi saltare un capitolo potrebbe significare per voi non capire piu' quanto segue (ovviamente questo non vale quando il testo vi rimanda ad altre parti del libro, come ad esempio puo' succedere per il glossario dei termini tecnici)

Se non avete capito qualcosa tornate sulla frase che non e' chiara e rileggetela. Se questa frase contiene dei termini che voi non conoscete probabilmente questi termini sono contenuti nel glossario che fa parte di questo libro e che si trova nel Capitolo 9.1.
Questo glossario non e' ovviamente un dizionario: contiene i termini tecnici normalmente sconosciuti a coloro che si avvicinano per la prima volta a un computer. Per altri termini che noi riteniamo di uso comune e che quindi non sono contenuti nel nostro glossario consultate un dizionario, ma non perdete il filo logico del discorso solo perche' non avete capito un termine. E' risaputo che quella che abbiamo appena enunciato e' una norma universalmente riconosciuta valida per fare di una lettura motivo di apprendimento.

ARRIVEDERCI AL PROSSIMO CAPITOLO.

CAPITOLO 1

BLA BLA BLA BLA BLA

CURA E MANUTENZIONE DEL SISTEMA

(ovvero cosa non dovete fare se volete che il vostro sistema funzioni a lungo...)

Il vostro COMMODORE 64 e' una sofisticata apparecchiatura elettronica cosi' come lo sono a diversi livelli il vostro televisore e il vostro impianto HI-FI. Come tale il vostro computer richiede certe cure e l'osservanza di alcune regole per l'uso: senza tutto questo potreste rovinare o danneggiare l'apparecchiatura sia pur involontariamente.
All'interno del Vostro COMMODORE 64, sotto la tastiera, dorme una complessa serie di circuiti elettronici quindi:
- non rovesciate liquidi sulla tastiera (alimentare correttamente il computer, come vedremo piu' avanti, non significa questo...)
- e' importante anche non far cadere sopra la tastiera la cenere della sigaretta: essa infatti penetra all'interno e puo' causare dei guasti
- anche la polvere che si accumula all'interno non contribuisce al buon funzionamento del vostro computer: riparatelo dalla polvere
- una corretta alimentazione del sistema e' pure essa importante: collegate la spina dell'alimentatore che viene fornito con la tastiera ad una presa di corrente da 220 volt (tutti i collegamenti sono ampiamente descritti piu' avanti)
- non fate cadere il vostro computer dal tavolo su cui probabilmente e' installato: l'urto potrebbe danneggiarlo.

Ma le cose piu' delicate con cui voi avrete a che fare dal momento in cui cominciate ad usare un computer sono i supporti magnetici, ossia le cassette per la registrazione di programmi e/o dati e i floppy disk (questi ultimi saranno chiamati familiarmente anche dischetti).
Una raccomandazione a proposito di supporti magnetici e' di usare solo cassette di ottima qualita', per evitare

che le testine del vostro registratore si sporchino in continuazione causandovi noie a non finire nella lettura e nella registrazione di dati e programmi. La buona qualita' e' requisito importante anche per i dischetti: non rischiate di perdere i vostri dati solamente per una piccola differenza di spesa...
- non avvicinate le cassette e i dischetti a fonti magnetiche. La vostra bella calamita a forma di ferro di cavallo non e' la sola fonte magnetica che si trova nel vostro studio: molte calamite (meglio chiamarli magneti) si mimetizzano infide nei posti piu' impensati: sapevate che la vostra cornetta del telefono contiene ben due magneti? e che qualsiasi altoparlante ha dietro di se' un grosso magnete? e che il fermacarte cosi'comodocheportaanchei fermagli tra le altre cose e' anche un magnete?. Anche un oggetto metallico (le forbici ad esempio) spesso e' magnetico. Tutti questi oggetti sono un reale pericolo per l'incolumita' dei dati memorizzati sui vostri supporti (a proposito non dimenticate l'altoparlante del televisore a cui avete collegato il vostro COMMODORE 64...)
- non esponete cassette e dischetti ai raggi solari diretti
- non conservateli a temperature superiori ai 50 gradi centigradi o inferiori ai 6 gradi
- non conservateli all'umidita'
- come per il computer, anche cassette e dischetti non devono essere esposti alla polvere a lungo: dopo l'uso riponeteli nella loro custodia (busta o scatola che sia)
- non maltrattateli: i dischetti non vanno piegati o cuciti con la normale cucitrice a un foglio di appunti. Se dovete spedirli per posta imballateli accuratamente (evitate il periodo natalizio e quello estivo e comunque... AUGURI!!!)

Come avrete potuto notare leggendo queste pagine non e' difficile aver cura del proprio computer: sono tutte norme elementari. Quindi: tenetele ben presenti sempre.

SINTASSI ED ABBREVIAZIONI USATE.

(ovvero cercate di capirmi...)

Nella lettura del testo, ed in particolare laddove il libro riporta dei programmi che voi dovrete scrivere sul computer, troverete simboli e/o abbreviazioni che vi invitano ad eseguire determinate operazioni.
Spesso e' difficile farsi capire scrivendo, soprattutto se ci si vuole rivolgere a persone inesperte e che non hanno ancora familiarita' con i termini e i modi di dire tipici di questo settore, cercate quindi anche voi di fare un piccolo sforzo e cercate di capire quello che a volte sembrera' incomprensibile ma che con un po' di riflessione e ragionamento diventera' piu'semplice.
E' molto importante che voi leggiate con attenzione le righe che seguono e, questo e' un nostro consiglio, fate un appunto su di un foglio da tenere a portata di mano per quando vi eserciterete sul computer.

ABBREVIAZIONI USATE	COSA DOVETE FARE
[RET]	premere il tasto con la scritta 'RETURN'
[CTRL] A	premere il tasto con la scritta 'CTRL' e, tenendolo premuto , premere il tasto della A
[SHIFT] A	premere il tasto con la scritta 'SHIFT' e, tenendolo premuto , premere il tasto della A
[C=] [SHIFT]	premere il tasto con il marchio COMMODORE e,tenendolo premuto, premere il tasto SHIFT
[CLR/HOME]	premere il tasto con la scritta 'CLR HOME'.

Quindi in pratica, quando troverete una scritta tra parentesi quadre e' sottointeso che significa premere il tasto del simbolo racchiuso tra parentesi quadre.
Se invece trovate due o piu' scritte tra parentesi

quadre o non di seguito, cio' significa che dovete premere tutti i tasti indicati contemporaneamente. I crampi alle dita sono optionals senza sovrapprezzo...
Anche altre piccole cose sono leggermente diverse dal nostro solito modo di agire, vedrete comunque argomento per argomento quali sono.
Ad esempio, diversamente che nell'uso comune, i simboli per eseguire operazioni matematiche sono leggermente diversi.
Ad esempio il segno di moltiplicazione che normalmente e' (X) ,diventa l'asterisco (*); quindi se noi scriveremo 6 x 4 non significhera' nulla, ma bensi' dovremo scrivere 6 * 4, che fa 24 anche per il vostro computer.
Quindi se vogliamo che il nostro COMMODORE 64 esegua una operazione aritmetica dobbiamo usare i seguenti simboli:

+ per fargli eseguire una somma
- per fargli eseguire una sottrazione
* per fargli eseguire una moltiplicazione
/ per fargli eseguire una divisione
↑ per fargli eseguire una elevazione a potenza

esempio: se il computer deve moltiplicare 6 per 4 dovremo scrivere:

6*4

se il computer deve sommare 3 piu' 3 dovremo scrivere:

3+3

e cosi' via...

Piu' avanti nella trattazione specifica di questi argomenti capiremo e vedremo in dettaglio l'uso e la funzione di questi simboli.

....SE TUTTO CAPITO VOLTA PAGINA BAGNANDO IL DITO.

CAPITOLO 2

UN PO' DI TEORIA

TERMINI SCONOSCIUTI

In queste righe vi diamo alcune spiegazioni circa i termini piu' importanti usati dagli addetti ai lavori.
Rimandiamo al glossario per altri termini che qui non citiamo e che vi risultano comunque di difficile comprensione.
Il miglior consiglio che posso darvi, a buon prezzo, e' quello di non spaventarvi mai troppo e di proseguire con caparbia perche' non e' poi cosi' difficile fare...l'esperto!!!

HARDWARE.

Dall'inglese Hard=duro e Ware=oggetto: sta ad indicare un sistema elettronico (come il vostro CBM64) per l'elaborazione dei dati, considerato sotto l'aspetto fisico, cioe' delle macchine e dei dispositivi che lo compongono fisicamente. Fanno parte dell'Hardware tutti i circuiti integrati, i fili, la tastiera e cosi' via. Ma, continuando nella scoperta del vostro computer, vi accorgerete che l'hardware (adesso sapete cos e', no?!...) senza il Software non puo' fare niente...

SOFTWARE.

Dall'inglese Soft=molle e Ware=oggetto: sta ad indicare l'insieme dei programmi che permettono ad un computer di eseguire le elaborazioni a cui e' stato destinato. Per chiarire ancora meglio la differenza tra HARDWARE E SOFTWARE potremmo paragonare il primo al nostro cervello ed il secondo alle idee che ci frullano in testa e ai concetti di cui facciamo uso normalmente per ogni attivita'.
Quando si dice 'fare del software' si indica la vera e propria attivita' di programmazione di un computer finalizzata a fargli svolgere un certo lavoro.
Il software (da ora in poi lo scriveremo minuscolo, vista la confidenza che abbiamo con lui ...) puo' essere scritto (o memorizzato) in diversi posti: su di una

cassetta magnetica, su di un dischetto oppure all'interno del vostro COMMODORE 64: qui dentro esso puo' risiedere nella ROM o nella RAM (cosa sono? - ve lo spieghiamo piu' avanti).

RAM.
Random Access Memory = memoria ad accesso casuale. Con il termine Ram si indica la memoria di lettura-scrittura che c'e' nel vostro COMMODORE 64. In questa memoria, che fisicamente e' costituita da chips (circuiti integrati), voi scriverete i vostri programmi e da questa memoria leggerete i dati elaborati dal computer il quale si serve pure lui di questa memoria per le sue elaborazioni. Quando spegnete il vostro COMMODORE 64, questa memoria si cancella completamente ed automaticamente.
La capacita' di memoria del vostro COMMODORE 64 e', cosi' come dice il suo stesso nome, pari a 64 Kbytes (si pronuncia chilobaits) ossia 65536 Bytes.

ROM.
Read Only Memory = memoria a sola lettura. A differenza della RAM, in questa memoria non possiamo scrivere niente, noi comuni mortali. Nel vostro COMMODORE 64 ci sono dei chips che contenevano dei programmi gia' quando voi lo avete acquistato e che voi non potete modificare: alcuni di questi chips, ad esempio, contengono il linguaggio basic, che voi conoscerete continuando le lettura di questo libro: ecco, questi chips si chiamano ROM. Un'altra differenza con la memoria RAM e' che quando spegnete il computer la ROM non si cancella, e' indelebile.

Questo e' il minimo indispensabile per non fare figuracce con gli amici quando parlate del vostro Computer. Ma non pensate di sapere tutto: questo e' solo l'inizio. Quando avrete letto tutto questo libro e sarete proprietari del vostro COMMODORE 64 da almeno qualche mese (e naturalmente a condizione che abbiate passato almeno tre notti insonni su di un programma che non voleva funzionare), parlerete come NOI DELL'INFORMATICA e i vostri amici vi guarderanno come Marziani. (Scrivo Marziani maiuscolo perche' vorrei evitare che uno di loro, durante la prossima invasione della Terra, si offenda leggendo queste righe e mi spruzzi di velenoso liquido verdastro...).

UNITA' PERIFERICHE (cosa sono?...)

Il computer non e' limitato alla sola parte ragionante: esistono altre cose che permettono alla macchina di capire o di farsi capire, le unita' periferiche.
Le unita' periferiche indispensabili che voi avete gia' a disposizione sono due: la tastiera per immettere dati o istruzioni e il video che permette al calcolatore di rispondervi.
Elenchiamo ora le principali unita' periferiche disponibili, chiarendone brevemente l'uso e la funzione.

A COSA SERVONO ?...

- DATASETTE C2N : il registratore a cassetta.
 assomiglia in tutto e per tutto ad un normale registratore a cassetta col quale ascoltate la musica. Come ormai sapete il computer ha dentro di se' una memoria chiamata RAM che si cancella allo spegnimento. Se noi scriviamo un programma e vogliamo conservarlo dobbiamo avere un supporto esterno al computer su cui andarlo a scrivere. Per far questo utilizzeremo il nostro registratore DATASETTE C2N. I dati (per dati intendo tutte le informazioni che passo al registratore) che ho memorizzato su di una normale audio cassetta potranno esssere poi restituiti al COMMODORE 64, sempre tramite il registratore, cosi come avviene per le canzonette che registriamo e possiamo poi riascoltare. Concludendo il nostro registratore e' la cosiddetta 'MEMORIA DI MASSA' cioe' quella memoria esterna al COMMODORE 64 che serve per tenerci archiavato una qualsiasi 'massa' di informazioni.

- DISK DRIVE VC1541 : il registratore/lettore di dischetti.
 Viene usato in alternativa al registratore ed assolve circa le stesse funzioni, ma con velocita' ed affidabilita' nettamente superiori. Tanto per capirci la differenza e' assimilabile a quella esistente tra un registratore a cassetta e un giradischi: per trovare l'ultima canzone incisa sul disco non siamo costretti a riavvolgere tutto il nastro, bensi' e' sufficiente spostare la testina del giradischi.L'altra grossa differenza sta nel fatto che su di un disco si possono gestire serie di dati anche in modo non sequenziale e quindi avere accesso veloce a grossi volumi di dati (sequenziale significa 'in sequenza', ossia uno di seguito all'altro).

- STAMPANTI

La stampante e' quella periferica che ci permette di avere su carta quanto normalmente vediamo a video ad esempio dati elaborati dal computer o liste di programma. Esistono tre tecniche diverse di stampa:

I - a matrice di punti (comunemente detta 'ad aghi') come le stampanti VC1525 o VC1526, che danno come risultato quella scrittura a punti tipica degli stampati fatti dai computer.Questa tecnologia permette di avere stampanti veloci a costi piuttosto bassi. La parte scrivente chiamata testina,ha dentro di se' una serie di aghi ,normalmente da 5 a 9, che spinti da una serie di elettromagneti nella giusta sequenza, battendo su di un nastro inchiostrato,formano il carattere desiderato.

II - a impatto, che e' il sistema tradizionale per esempio quello della macchina per scrivere a margherita. La qualita' di stampa guadagna a scapito della velocita'. Viene usata generalmente da quanti hanno bisogno di un'alta qualita' di stampa (come i notai per i loro Atti ecc.).Spesso questo tipo di stampanti ha costi alti e velocita' bassa.Questa tecnica di stampa e' la piu' vecchia e tradizionale.Un martelletto ,pilotato da un elettromagnete ,batte su di una matrice di plastica o metallo, dove vi e' inciso il carattere,che battendo sulla carta con interposto del nastro inchiostrato da' la stampa del carattere (proprio come faceva Gutenberg).

III- termica, che produce la scrittura scaldando della carta termosensibile. Stampanti che usano questa tecnica hanno solitamente un basso costo d'acquisto e sono silenziosissime, ma costa poi un po' di piu' la carta, visto che si tratta di carta speciale termosensibile.

ALTRE...

Anche il Joistick (la cloche di comando per giochi) o le paddles (le manopole per i giochi) che voi sicuramente conoscete sono unita' periferiche, che come tali interagiscono col vostro COMMODORE 64.

Esistono altre unita' periferiche di uso meno comune ad esempio il MODEM (per chi ha visto WAR GAMES...), ma non riteniamo opportuno parlarne diffusamente in questa sede.

ARRIVEDERCI ALLA PROSSIMA PUNTATA DI DALLAS...

CAPITOLO 3

COMINCIAMO AD USARLO

COLLEGAMENTO DEL SISTEMA.

Proviamo a collegare tutti i pezzi del nostro sistema. Prendiamo per prima cosa l'alimentatore, che e' quella scatola da cui partono due cavi, uno che termina con una spina da inserire nella rete (220volt) e l'altro con una spina tonda con 5 poli. L'alimentatore non contiene altro che un trasformatore e va collegato ad una presa a 220 volt. Evitate di usare castelli e torri di riduzioni e spine triple onde evitare falsi contatti e spiacevoli sorprese. Detto questo collegatelo ad una presa di corrente.

FIG. 1. Il vostro acquisto.

Accertatevi che l'interruttore posto sulla destra del vostro COMMODORE 64 (FIGURA 2 particolare 2) sia in posizione OFF (spento quindi). Ora collegate la spina a 5 poli alla presa che si trova di fianco all'interruttore sul COMMODORE 64 (FIGURA 2 particolare 1).

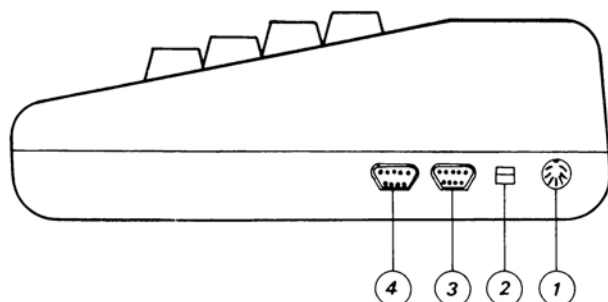



FIG.2. Vista del fianco destro con presa di alimentazione, interruttore e porte joistick

Le altre due prese, che trovate sul fianco del COMMODORE 64 (particolare 3 e 4 della FIGURA 2) serviranno per collegare i Joisticks o le Paddles per i giochi. Queste due prese sono contrassegnate dalle scritte 'PORT 1' e 'PORT 2', questo lo specifichiamo perche' alcuni giochi chiedono di collegare il comando alla porta 1 o alla porta 2.
Prendete il cavo per il collegamento al televisore: da un lato e' provvisto di una spina simile a quella usata per il collegamento degli impianti HI-FI e dall'altro lato di una normale spina per antenna TV. Quest'ultima va collegata al posto del cavo antenna nel vostro televisore, l'altra spina va invece collegata sul retro del vostro COMMODORE 64 nell'apposita presa (FIGURA 3 particolare 7).
Prendete ora il vostro registratore a cassette C2N (se lo avete) e collegate il cavo che ne fuoriesce al vostro COMMODORE 64 nell'apposita presa posteriore: riferitevi

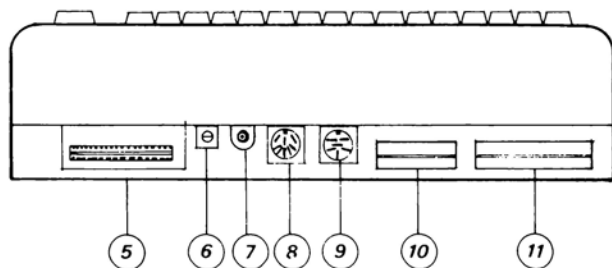



FIG.3. Vista posteriore con prese per connessioni varie

alla FIGURA 3 Particolare 10. Attenzione: questa spina va inserita esercitando una certa pressione, ma rispettando la tacca di riferimento, che ha una corrispondenza nella presa posteriore del computer.
A questo punto accendete il TV e di seguito il COMMODORE 64 spostando l'interruttore in posizione ON (acceso!).
Il LED (diodo luminescente, insomma la spia rossa che e' illustrata in FIGURA 6 Particolare 1) si illuminera' indicando l'accensione del sistema. Per ora sul vostro TV non si vedra' ancora niente.

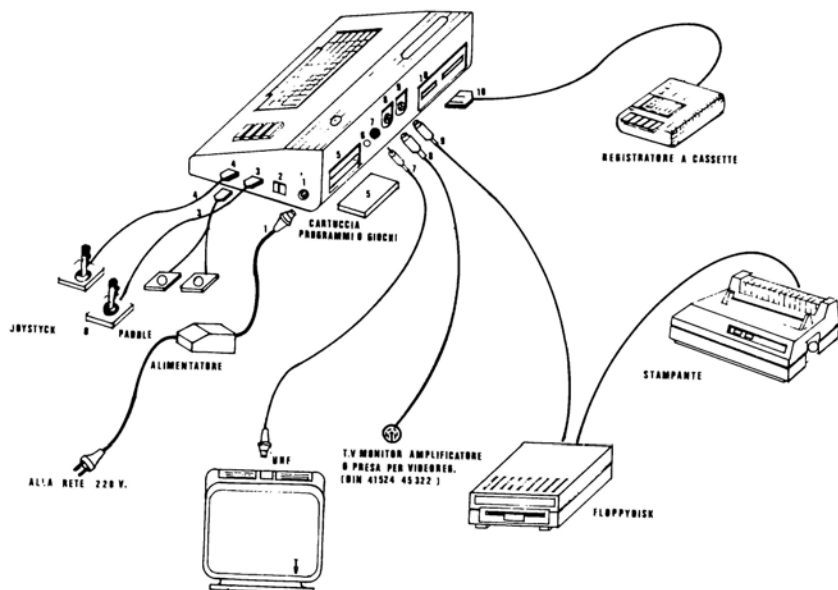



FIG.4. Schema di alcune delle periferiche collegabili direttamente al sistema

Cercate ora di sintonizzare il TV col computer cosi' come se cercaste una emittente televisiva: generalmente il COMMODORE 64 lo trovate intorno al canale 36 in UHF. Sul video con uno sfondo azzurro chiaro ed un riquadro centrale blu dovrete vedere le seguenti scritte:

```
   **** COMMODORE 64 BASIC V2 ****
64K RAM SYSTEM  38911 BASIC BYTES FREE

READY
```

Aggiustate il contrasto e la luminosita' tramite i comandi del vostro televisore sino ad ottenere una lettura chiara ed il colore del riquadro centrale blu (ovviamente se siete muniti di un TV in bianco e nero non vedrete altro che grigi in diverse gradazioni...).
Se non ottenete questi risultati ripartite dall'inizio di questo paragrafo eseguendo nuovamente tutte le operazioni in sequenza.
Per il corretto collegamento delle altre unita' periferiche fate riferimento al capitolo specifico dell'unita' che intendete collegare.
Ricordatevi in ogni caso che qualsiasi collegamento o scollegamento deve essere eseguito con l'interruttore

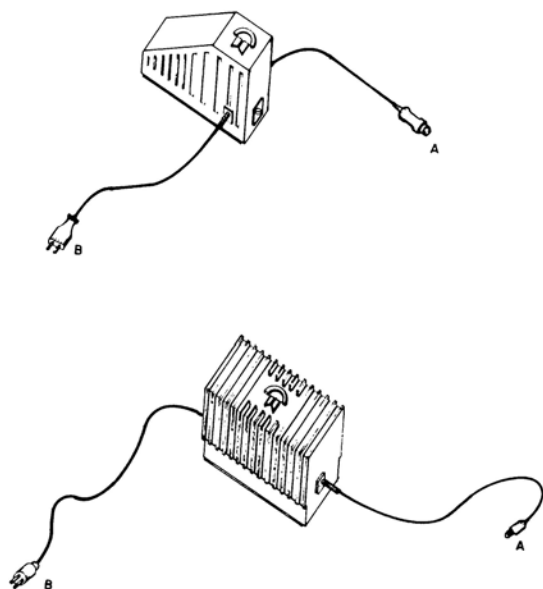



FIG.5. Alimentatori del Commodore 64

del vostro COMMODORE 64 in posizione OFF (spento) per evitare possibili guasti.
Se volete inserire una Cartridge, o cartuccia, contenente un programma, eseguite il collegamento della Cartridge a computer spento, infilandola con decisione nel vano posteriore apposito (FIGURA 3 particolare 5).

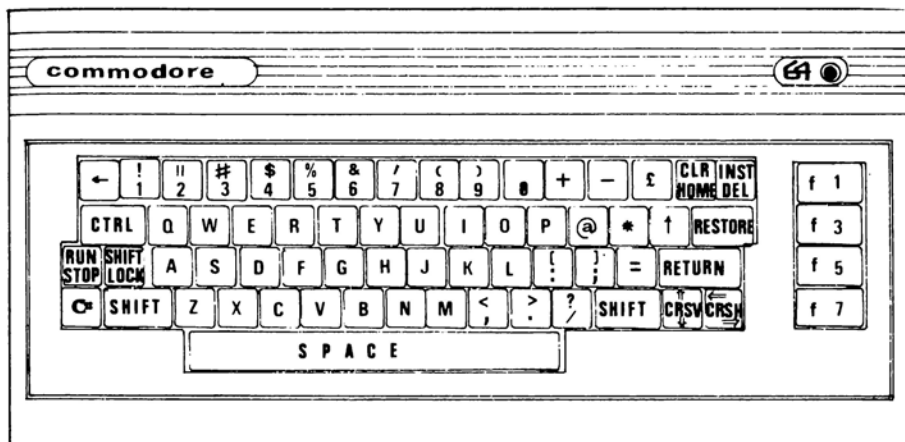



FIG.6. Tastiera e led di accensione

A proposito, ora che avete acceso il vostro COMMODORE 64, non spegnetelo: leggendo il paragrafo successivo sarete invitati a fare alcune prove che vi permetteranno di capire la funzione dei diversi tasti. Finalmente ci mettiamo su le mani, eh ?!...

CONOSCENZA E USO DELLA TASTIERA

E' molto importante prendere confidenza con la tastiera poiche' e' l'unico mezzo che voi avete per colloquiare col vostro computer, impartirgli ordini o inserire dati.
Sotto molti aspetti la tastiera e' simile a quella di una comune macchina per scrivere. Essa comprende in piu' alcuni tasti particolari di cui parleremo ora.
E' importante chiarire anzitutto il significato di una parola che troverete nelle pagine successive e che e' 'cursore'. Essa sta ad indicare quel quadratino lampeggiante che vedete sul video e che indica il punto in cui apparira' quanto scrivete con la tastiera.
Analizziamo ora i tasti speciali che vedete in FIGURA 6.

RETURN.

E' il tasto che permette di confermare cio' che avete scritto a video e dice al vostro COMMODORE 64 "...ora tocca a te". Come risultato da' anche il ritorno a capo del cursore sulla riga successiva: provate a premerlo senza aver premuto nessun altro tasto e vedrete che il cursore va a capo sulla linea successiva.
A questo tasto faremo piu' volte riferimento in questa sede con:
[RET]
che significa premere il tasto RETURN.
Il nome originale della funzione propria di questo tasto e' "CARRIAGE RETURN"; infatti in alcuni testi e nel gergo ci si riferisce a questo tasto con la dicitura "CR".

SHIFT

Questo tasto ha funzione simile all'omonimo tasto della macchina per scrivere. In quest'ultima infatti esso serve per generare le lettere maiuscole o minuscole a seconda che sia stato o meno premuto.
All'accensione il vostro COMMODORE 64 scrive a caratteri maiuscoli. Per poter scrivere in modo minuscolo/maiuscolo occorre premere simultaneamente il tasto C= e il tasto SHIFT. Da quel momento la normale scrittura sara' in minuscolo e useremo il tasto SHIFT contemporaneamente al tasto della lettera che vogliamo scrivere in maiuscolo. Per ritornare al modo 'solo maiuscole' ripremere [SHIFT] [C=]. Il tasto SHIFT va usato anche per raggiungere i simboli posti sulla parte superiore dei tasti su cui compaiono due simboli: ad esempio / e ? sono sullo stesso tasto, 2 e " sono sullo

stesso tasto. Provate a premere il tasto con 2 e " e vedrete che sul video compare il numero 2. Per scrivere a video le virgolette occorrera' premere il tasto SHIFT e (tenendolo premuto) premere il tasto col numero 2: in questo modo otterremo quanto voluto. La stessa cosa per tutti gli altri tasti a due funzioni. Non confondiamoci coi simboli grafici posti sulla parte frontale di alcuni tasti di cui parleremo piu' avanti).

SHIFT
LOCK.

Ha le stesse funzioni del tasto SHIFT, ma puo' rimanere permanentemente premuto grazie ad uno speciale interruttore. Provate a premerlo: sentirete un 'click' ed il tasto rimarra' piu' basso rispetto al livello degli altri, ripremetelo e il tasto si rialzera' col solito 'click'.
Facile no?..Fa 'click-click'!..

C=

E' il tasto col marchio Commodore posto nell'angolo in basso a sinistra sulla vostra tastiera. Una funzione del tasto C= consiste nella capacita', se premuto contemporaneamente al tasto [SHIFT], di passare dal modo solo maiuscole al modo minuscole/maiuscole e viceversa, come abbiamo descritto sopra.
Una seconda funzione e' quella che permette di scrivere i simboli grafici raffigurati sulla parte frontale dei tasti e solo quelli posti sulla sinistra di ogni tasto. Provate a premere il tasto C= e, tenendolo premuto, premete il tasto della lettera V: apparira' un quadratino bianco corrispondente al simbolo grafico di sinistra. Per riprodurre invece i simboli grafici posti sulla destra dei tasti occorre essere in modo 'solo maiuscole' e usare il tasto SHIFT contemporaneamente al tasto scelto.
L'ultima funzione che elenchiamo e' la seguente:

cambiare il colore di scrittura dei caratteri. Premete [RET]. Provate a premere [C=] 3 (vuol dire premere il tasto C= e di seguito, senza rilasciarlo, premere anche il tasto della cifra 3): tutti i caratteri che scriverete di seguito saranno visualizzati in colore rosso.Provate a scrivere il vostro nome !. Ripetete l'operazione con tutti i colori che vedete scritti abbreviati in lingua inglese sul fronte dei tasti numerici.

Riportiamo le abbreviazioni usate sulla tastiera:

BLK = BLACK - NERO
VHT = WITHE - BIANCO
RED = RED - ROSSO
CYN = CYAN - AZZURRO
PUR = PURPLE- VIOLETTO
GRN = GREEN - VERDE
BLU = BLUE - BLU
YEL = YELLOW- GIALLO

CTRL

Questo tasto, la cui dicitura e' l'abbreviazione di CONTROL, serve per accedere a funzioni particolari. Premendolo da solo non si sortisce effetto alcuno. Va sempre usato in congiunzione con altri tasti e troverete la spiegazione delle diverse funzioni nel corso della lettura di questo libro.

RESTORE

Anche questo tasto non ha funzioni se usato da solo. Premendolo unitamente al tasto RUN/STOP si interrompe l'esecuzione di qualsiasi programma, si cancella il video (ma non la memoria) e si ripristinano le condizioni di partenza (modo solo maiuscole e colore dello sfondo azzurro con riquadro centrale blu). Corrisponde praticamente al RESET o al BREAK di altri computer.

RUN
STOP.

Premendolo, si interrompe l'esecuzione di un programma (salvo in alcune condizioni particolari, ad esempio quando il computer attende dei dati da tastiera: in questo caso usare [RUN/STOP] [RESTORE]).
[RUN/STOP] usato da solo non cancella il video ne' la memoria.
Usato in congiunzione col tasto SHIFT sostituisce il comando LOAD, che serve a far leggere al computer un programma dalla cassetta (non usatelo ora: vedete il paragrafo dedicato al registratore).
Provate ora :
premete il tasto RUN/STOP e, senza rilasciarlo, premete (con un piccolo colpetto, come se martellaste il torace di un paziente) il tasto RESTORE: e' sparito tutto sul

video ?? .. Se ha funzionato OK, se no riprovate: deve funzionare...

CLR
HOME.

Premendolo, porta il cursore (quel quadratino lampeggiante che indica sul video il punto in cui si andra' a scrivere) alla posizione in alto a sinistra. Questa e' la funzione di 'HOME'.
Usato in congiunzione col tasto SHIFT esegue invece la funzione 'CLEAR', che oltre a riportare il cursore in alto a sinistra cancella anche tutto quanto scritto a video (ma non la memoria), cosi' come abbiamo visto piu' sopra con i tasti [RUN/STOP] [RESTORE].

TASTI
CRSR

Ci sono due tasti CRSR (abbreviazione di cursore). Uno con freccia in alto e freccia in basso, l'altro con freccia a destra e freccia a sinistra. Questi tasti servono per muovere il cursore sul video nelle quattro direzioni. Precisamente il tasto CRSR → sposta il cursore lateralmente di un carattere: a destra se premuto da solo, a sinistra se premuto contemporaneamente al tasto SHIFT. Il tasto CRSR↕ sposta invece il cursore in alto o in basso di una riga e precisamente in basso se premuto da solo, in alto se premuto contemporaneamente al tasto SHIFT.

Li chiameremo, da ora in poi, CRSV il tasto CRSR con le due frecce verticali e CRSH quello con le frecce orizzontali.

INST
DEL.

Provate a scrivere con la tastiera il vostro nome (non battete [RET]); ora premete il tasto INST/DEL e l'ultimo carattere che avete digitato sara' cancellato e sostituito dal cursore lampeggiante. La sua funzione e' quindi quella (dall'inglese DELETE=cancella) di cancellare il carattere a sinistra del cursore e quindi correggere eventuali errori di battitura.
Usato in congiunzione col tasto SHIFT serve a inserire uno spazio vuoto in mezzo a una parola (funzione INST=INSERT) e sempre comunque alla sinistra del cursore. Battete [RET]. Se avete premuto tutti i tasti

che vi abbiamo indicato finora, sicuramente col [RET] il vostro COMMODORE 64 scrivera' a video:

?SINTAX ERROR

e riportera' il cursore a capo. Non preoccupatevi, questo significa solo che quanto noi abbiamo digitato non ha per lui alcun significato come comando e quindi ci risponde con una segnalazione di errore di sintassi e si predispone a ricevere qualche ordine piu' sensato...
Ora scrivete MAMA, che non e' un'istruzione del linguaggio basic, ma e' semplicemente la parola MAMMA con una emme di meno. Premete [SHIFT] [CRSH] per due volte cosi' che il cursore sia posizionato sulla 'M'; premete ora [SHIFT] [INST/DEL] e vedrete comparire uno spazio a sinistra della 'M' con il cursore lampeggiante. Ora battete il tasto 'M' ed ecco che siete riusciti a correggere l'errore inserendo la lettera mancante. Con il tasto [CRSH] potete ora portare il cursore alla destra della parola 'MAMMA'.

f1/f2/f3/f4/f5/f6/f7/f8

Sono comunemente indicati col termine di 'TASTI FUNZIONE'.
Premendo questi tasti sul video non compare nulla. Questo non significa che nessun messaggio sia arrivato all'orecchio del vostro COMMODORE 64. Infatti questi tasti generano dei comandi che non hanno effetto alcuno se non indicato da un programma che li gestisca. Se in un programma noi prevedessimo l'uso di uno di questi tasti (f1, ad esempio) per sparare a un alieno sara' il programma che dice al computer che il messaggio che gli arriva mediante il tasto funzione 1 deve sortire l'effetto dello sparo all'alieno. Quindi, al di fuori di un programma, non hanno alcuna funzione, mentre se previsto dal programma possono essere impiegati nei modi piu' disparati.
Precisiamo comunque che per raggiungere i tasti f2,f4,f6 ed f8 occorre usare contemporaneamente al tasto funzione che reca frontalmente la dicitura corrispondente, il tasto SHIFT.

IL REGISTRATORE

Per collegare il registratore al vostro COMMODORE 64 bisogna solamente collegare il cavo che esce dal registratore nella presa posteriore del computer (FIGURA 3 posizione 10) ,facendo attenzione alla tacca di riferimento, che non permette di inserire l'apposita spina al contrario.
Ricordiamo che tutte le operazioni di collegamento o scollegamento di cavi, spine e o periferiche al vostro COMMODORE 64 deve essere fatto a computer spento.
Poniamo ora il caso di aver comperato una cassetta con un programma o di avere un nostro programma precedentemente scritto,e di volerlo caricare nella memoria del nostro COMMODORE 64. Come fare? :
-accendete il COMMODORE 64
-sintonizzate il TV
-inserite la cassetta nel registratore con il lato registrato posto in alto.
-riavvolgete tutto il nastro con il tasto del registratore contrassegnato REWIND, quando il nastro e' riavvolto completamente premete il tasto STOP (sempre sul registratore)
-digitate sulla tastiera:

LOAD "NOME PROGRAMMA" [RET]

dove NOME PROGRAMMA e' il nome del programma che desiderate caricare; se non sapete o non vi ricordate il nome del programma e' sufficiente battere:

LOAD [RET]
(omettendo quindi il nome del programma)

A questo punto il COMMODORE 64 risponde :

PRESS PLAY ON TAPE

e voi, obbedienti, premete il tasto PLAY sul registratore. Il video diventera' completamente bianco; dopo alcuni secondi sullo schermo comparira':

FOUND (NOME PROGRAMMA)

dove per (NOME PROGRAMMA) si intende il nome del programma che il registratore ha trovato. Se avete fatto seguire al comando [LOAD] il nome del programma, e questo coincide con quello trovato sulla cassetta, il caricamento avviene automaticamente dopo alcuni secondi (se non volete aspettare, potete evitare questa attesa

premendo il tasto C= e il caricamento inizia subito). Il video diventera' ancora bianco ed il registratore, che nel frattempo si era fermato, ripartira' per poter caricare cosi' l'intero programma. Se avete battuto solo LOAD (senza nome, quindi) il caricamento avviene come sopra (percio': visualizzazione del programma trovato e inizio caricamento dopo alcuni secondi). Se il nome del programma trovato e' diverso da quello richiesto da voi, il COMMODORE, dopo aver segnalato a video con FOUND (NOME PROGRAMMA) il titolo del programma incontrato, prosegue alla ricerca del prossimo memorizzato sulla cassetta (un trucco per sapere quali programmi ci sono su una cassetta? Battete dopo il comando LOAD un nome/programma assurdo -che presumete non esista- e lasciate che il COMMODORE esplori tutta la cassetta: alla fine troverete sul video l'elenco dei programmi trovati).
Quando il COMMODORE 64 rispondera'

READY

(questo puo' succedere anche dopo 5 o 6 minuti dipende dalla lunghezza del programma)

avrete in memoria il programma, che con l'istruzione

RUN [RET]

verra' eseguito. Attenzione: questa operazione di caricamento di un programma azzera il contenuto della memoria, quindi, se voi avevate in memoria qualche cosa nel momento in cui date il comando LOAD, questo qualcosa viene cancellato.
Come abbiamo gia' accennato nel paragrafo riguardante la tastiera, si puo' dare il comando per caricare un programma in memoria anche tramite i tasti [SHIFT] e [RUN/STOP] premuti contemporaneamente: quando il COMMODORE segnala il FOUND o ritrovamento di un programma, occorre premere il tasto C= ed il programma viene caricato ed eseguito senza bisogno di battere il comando RUN [RET].
Per scrivere invece un programma su di una cassetta (anche se per ora non ne sapete ancora fare) le operazioni da eseguire sono:

-prendere una cassetta pulita (cioe' non lavata con il detersivo che da' il bianco piu' bianco) e riavvolgerla completamente
-digitare:

SAVE "NOME PROGRAMMA" [RET]

Dove NOME PROGRAMMA e' il nome che volete dare al vostro programma che non puo' essere piu' lungo di 16 caratteri. Il COMMODORE 64 rispondera':

PRESS PLAY AND RECORD ON TAPE

Premete contemporaneamente il tasto RECORD e il tasto PLAY, il video diventera' bianco e il registratore si mettera' in moto. Quando il programma sara' salvato (cioe' memorizzato) sul vostro nastro il COMMODORE 64 rispondera':

READY

indicando che il tutto e' stato eseguito e che il vostro programma, oltre che in memoria, e' anche registrato sul nastro (ora premete il tasto STOP sul registratore).

Esiste anche la possibilita' di verificare se il programma che avete appena memorizzato sul nastro e' stato scritto correttamente dal registratore mediante cio' che si chiama VERIFY (verifica)
Facciamo un esempio:
dopo avere acceso il vostro COMMODORE 64 battete:

10 PRINT "PROVA" [RET]

Questo minuscolo programma, che non ha alcun significato, vi servira' come esempio per fare una prova generale.
Prendete una cassetta nuova e inseritela nel vostro registratore. Ora riavvolgete tutto il nastro col tasto REWIND e digitate:

SAVE "PROVA" [RET]

IL COMMODORE 64 rispondera':

PRESS PLAY AND RECORD ON TAPE

premere contemporaneamente i tasti RECORD e PLAY sul registratore. Il video si cancella e il registratore si avvia per memorizzare il programma. Quando il COMMODORE 64 rispondera' con la scritta:

READY

riavvolgete il nastro premendo il tasto REWIND. Quando tutto il nastro sara' riavvolto, premete il tasto di STOP e battete:

VERIFY "PROVA"

Il COMMODORE 64 rispondera'

PRESS PLAY ON TAPE

premete ora il tasto PLAY del registratore. A questo punto, se l'operazione di scrittura e' stata eseguita con successo, dopo qualche secondo apparira' la scritta OK.
Cosa e' successo? Il COMMODORE 64 ha confrontato il programma che ha in memoria con quello scritto sul nastro: se le due cose coincidono, l'operazione di memorizzazione e' andata bene e ve lo segnala con un OK.
Ora fate un 'miracolo': rileggete quanto avete appena memorizzato sulla cassetta.
Spegnete il computer tramite l'apposito interruttore ON/OFF.
Dopo alcuni secondi riaccendetelo e il vostro programma e' sparito. Nota: dopo lo spegnimento occorre sempre attendere tre o quattro secondi prima di riaccendere.
Riavvolgete il nastro della cassetta premendo il tasto REWIND sul registratore.
A riavvolgimento ultimato, premete il tasto STOP del registratore.
Battete ora sulla tastiera il seguente comando:

LOAD "PROVA" [RET]

Sul video apparira' la scritta

PRESS PLAY ON TAPE

A questo punto premete il tasto PLAY sul registratore e attendete fino a che sul video non comparira' la scritta

FOUND PROVA

Premete il tasto C= per confermare la lettura del programma "PROVA" e rimanete in attesa.
A lettura ultimata il COMMODORE fermera' automaticamente il nastro e scrivera' a video

READY

seguito dal cursore lampeggiante.

Ora il programma si trova nella memoria del COMMODORE 64 e potete controllarlo battendo

LIST [RET]

A video ricomparira' di seguito a questo ultimo comando l'istruzione che avevate memorizzato sulla cassetta.

Che fatica, eh?!...

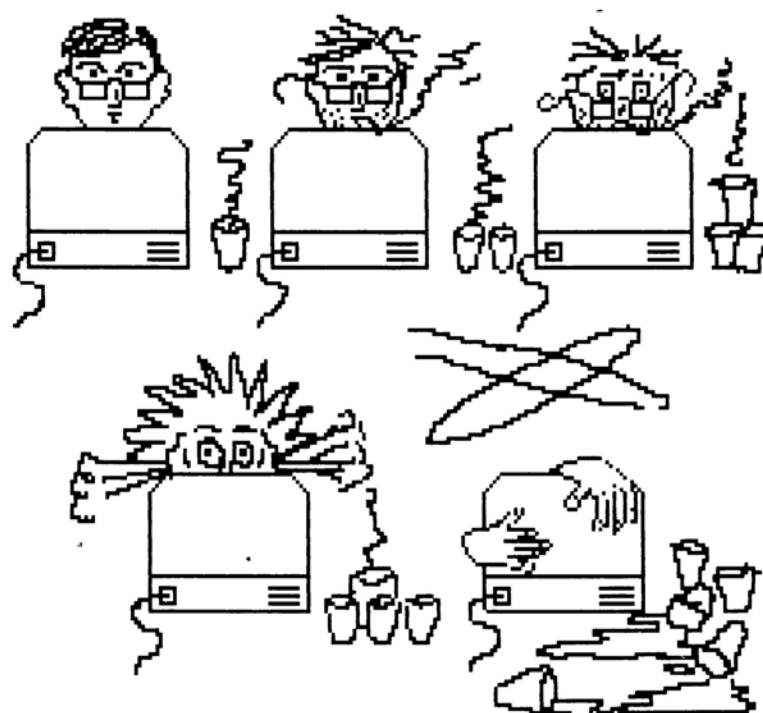



FIG.7. E questo non e' che l'inizio.....

EDITOR
(non c'entra con editore...)

Proviamo ora a muovere il cursore sul video usando i tasti [CRSV] e [CRSH] in congiunzione col tasto [SHIFT] o da soli: il cursore puo' attraversare il video in tutte le direzioni.
Questo facilita la scrittura e la correzione di un programma: l'insieme dei comandi che svolgono la funzione di movimento del cursore a video viene definito EDITOR. L'editor del vostro COMMODORE 64 e' ben fatto, in quanto consente con estrema facilita' di svolgere i compiti per i quali e' stato congeniato.
Provate a premere [SHIFT] e (senza rilasciarlo) [CRSV] e il cursore si muovera' verso l'alto; per muoverlo verso il basso basta invece premere [CRSV].
Provate ora a premere i tasti [SHIFT] e [CRSH] per muovere il cursore a sinistra; per muoverlo a destra bastera' premere [CRSH].
Ora immedesimatevi col cursore e fate un giretto sul video esplorandolo in tutte le posizioni, cosi', tanto per familiarizzare coi tasti [CRSV] e [CRSH].
A cosa puo' servire muoversi sul video?
Provate a battere i seguenti tasti:
[RUN/STOP] [RESTORE]
il video si cancella e il cursore si posiziona nell'angolo in alto a sinistra.
Scrivete ora
BABBO
(si', proprio come BABBO NATALE...)
se invece di BABBO aveste voluto scrivere MAMMA, senza l'esistenza dell'EDITOR vi sarebbe stato impossibile tornare indietro.
Provate ora cosi':
- battete per 5 volte i tasti [SHIFT] e [CRSH] fino a che il cursore si sara' posizionato sulla prima B di BABBO
- battete ora MAMMA ed otterrete che si sovrapponga a BABBO.
Questo esempio vi ha mostrato a cosa puo' servire spostare il cursore a video.
Ci sono altri tasti che possono servire per l'EDITOR, che avete gia' incontrato nella descrizione dettagliata della tastiera. Di seguito sono rappresentati i tasti coinvolti nell'EDITOR.

Ora siete in grado di correggere qualsiasi cosa ci sia a video.
Ma se volete che il cursore si muova docile per ogni dove anche da programma, dovete sapere che i comandi visti sopra si possono usare anche in differita, ossia come istruzioni di un programma.

CRSR sú	(CRSV)	CRSR	VERTICALE IN SÚ	●
CRSR giũ	(CRSV)	CRSR	VERTICALE IN GIÚ	Q
CRSR sin.	(CRSH)	CRSR	ORRIZZ. a sinistra	|
CRSR destr.	(CRSH)	CRSR	ORRIZZ. a destra	]
CLR		CLR/	CANCELLA VIDEO	♥
HOME		/HOME	CURS. INIZIO VIDEO	S
INST		INST/DEL	INSERIMENTO	|

Non fatevi spaventare dalle difficolta' che probabilmente incontrate a seguire queste spiegazioni: quando sarete piu' pratici tornerete a rileggere attentamente (speriamo...) questo paragrafo, visto che l'utilita' dei comandi di cui stiamo parlando si rivelera' insostituibile a mano a mano che procederete nell'apprendimento della programmazione.
Quindi dicevamo che i comandi dell'EDITOR si possono dare anche da programma: vediamo come.
Anzitutto non dovete mai dimenticare che per inserire tali comandi in una istruzione, essi devono essere compresi tra virgolette o apici ("); chiameremo questo: 'modo virgolette'.
Se provate a battere uno dei tasti [CRSR] tra virgolette, otterrete il simbolo grafico che rappresenta il tasto che avete premuto.
Se tale comando e' inserito in un programma a mo' di istruzione l'effetto non sortisce immediato come quando premete uno dei tasti [CRSR] ed il cursore si muove docilmente, bensi' occorre dare il via al programma (con RUN) perche' tale comando venga eseguito.

Un esempio e' d'obbligo:
Battete ora
[RUN/STOP] [RESTORE]
NEW [RET]

(nota bene i simboli grafici dopo la parola CIAO: si ottengono premendo i tasti [SHIFT] [CRSH], ma solo quando si e' all'interno delle virgolette)

ora scrivete

RUN [RET]

che serve a far funzionare il programma (in questo caso composto dalla sola riga 10, ma pur sempre programma)

Il vostro COMMODORE 64 rispondera' cosi':

MIAO
READY

In effetti il COMMODORE ha eseguito esattamente l'istruzione in sequenza:

- ha scritto CIAO eseguendo il comando PRINT che significa 'STAMPA'
- ha eseguito i comandi di CRSH, muovendo quindi il cursore verso sinistra di 4 posizioni e portandolo sopra la lettera C
- infine ha stampato (poiche' il comando prima delle virgolette e' sempre il solito PRINT) la lettera M sovrapponendola alla C

Chiaro?

Parliamo ora dell'uso in 'modo virgolette' (ossia all'interno di un programma) degli altri comandi:

HOME serve per portare il cursore in alto a sinistra senza cancellare il video

CLR serve a cancellare il video e portare il cursore in alto a sinistra

INST serve per inserire uno spazio vuoto

DEL serve per cancellare un carattere alla sinistra del cursore

E' possibile usare nel 'modo virgolette' anche i comandi per il cambiamento del colore, dei caratteri e del reverse e delle funzioni.
La procedura per l'utilizzo e' la stessa e di seguito la figura vi indica quali tasti sono da usare e quali simboli appariranno sul video (eventualmente consultate anche le tavole in appendice).

COSA BATTERE		COLORE	COSA APPARE
CTRL	1	NERO	■
CTRL	2	BIANCO	[illegible]
CTRL	3	ROSSO	[illegible]
CTRL	4	CYAN	[illegible]
CTRL	5	PORPORA	[illegible]
CTRL	6	VERDE	[illegible]
CTRL	7	BLU	[illegible]
CTRL	8	GIALLO	[illegible]
C=	1	ARANCIO	[illegible]
C=	2	MARRON	[illegible]
C=	3	ROSSO CHIARO	[illegible]
C=	4	GRIGIO SCURO	[illegible]
C=	5	GRIGIO	[illegible]
C=	6	VERDE	[illegible]
C=	7	AZZURRO	[illegible]
C=	8	GRIGIO CHIARO	[illegible]

Provate a mettere tutti questi comandi tra virgolette dopo un'istruzione di PRINT e controllate che i simboli che appaiono siano corrispondenti a quelli della figura.
Avrete modo di apprendere in maniera piu' efficace l'uso di questi comandi approfondendo la vostra conoscenza del basic mediante la lettura dei capitoli successivi.

CAPITOLO 4

IL LINGUAGGIO DEL COMPUTER

IL BASIC
(acceso e pronti a collaborare)

E' il linguaggio che vi permettera' di parlare e far ragionare il vostro computer.Come tale richiedera' il rispetto di un certo numero di regole ortografiche e logiche, necessarie per comunicare correttamente al vostro COMMODORE 64 che cosa desiderate venga realmente eseguito.
Ogni comando puo' essere dato al COMMODORE 64 :

1) direttamente
2) sotto forma di istruzione da eseguire all'interno di un programma.

1- Battete senza commettere errori

```
PRINT "QUESTA E' LA PRIMA PROVA"
```

Ora battendo [RET] il calcolatore accetta come comando cio' che e' scritto sul video nella riga in cui lampeggia il cursore e lo esegue direttamente stampando sul video tutto cio' che e' contenuto tra le virgolette e cioe':

```
==> QUESTA E' LA PRIMA PROVA
==>
==>  READY     ( CHE VUOL DIRE SONO PRONTO )
```

2- Battete invece :

```
100 PRINT "QUESTA E' LA PRIMA PROVA"
```

seguito da [RET] :

(AVETE SCRITTO LA VOSTRA PRIMA ISTRUZIONE DI PROGRAMMA)

Il cursore si sposta a capo sulla riga successiva e non avviene nulla di visibile, ma il COMMODORE 64 ha memorizzato il comando non solo sul video ma anche in

una parte della memoria riservata ai programmi (vi ricordate la RAM ? NO, allora tornate a pag. 6 e state fermi un giro!).

Battete ora [SHIFT] e [CLR/HOME] che serve per pulire il video, ora il COMMODORE 64 ha dimenticato il primo comando (quello diretto) ma il secondo esiste ancora e lo si puo' vedere battendo:

LIST [RET]

apparira':

```
==> 100 PRINT "QUESTA E' LA PRIMA PROVA"
==> READY
```

Quindi il comando in esecuzione differita, cioe' preceduto dal numero, non e' scomparso.
Ora se voglete, finalmente, l'esecuzione del programma dovrete battere:

RUN

e poi battere [RET] , apparira':

```
==> QUESTA E' LA PRIMA PROVA
==> READY
```

La differenza tra i due casi or ora visti, e' dovuta alla presenza di un numero, denominato numero di linea messo prima di ogni istruzione, che permette al COMMODORE 64 di capire (e ci riesce, LUI !) se l'ordine andra' eseguito subito o memorizzato nella RAM con le altre istruzioni da eseguire successivamente.
E' comprensibile che non abbiate capito molto, tra poche righe capirete con un altro esempio:

I NUMERI DI LINEA

- devono essere interi e non preceduti da segni
- devono essere compresi tra 0 e 63999 .
- danno l'ordine in cui devono essere eseguite le istruzioni
- il COMMODORE 64 ordina le linee di programma secondo i numeri di linea anche se le stesse vengono scritte in tempi o posti diversi
- I numeri di linea si devono scrivere senza virgole,punti o altri segni, altrimenti il computer puo' capire male e farvi pensare che abbia sbagliato.

- qualsiasi segno non numerico appartenente al numero di linea suddivide il numero stesso in due parti: la prima funziona come numero di linea, la seconda viene interpretata come un'istruzione (probabilmente errata)

Esempio:

```
-200 PRINT "SECONDA PROVA"     [RET]
35,8 PRINT "TERZA PROVA"       [RET]
108000 PRINT "QUARTA PROVA"    [RET]
5.000 PRINT "QUINTA PROVA"     [RET]
```

la prima e la terza linea provocano una segnalazione di errore non appena premuto [RET], non vengono accettate come istruzioni ne' memorizzate, la seconda e la quarta vengono invece memorizzate come:

```
35 ,8 PRINT "TERZA PROVA"
5  .000 PRINT "QUINTA PROVA"
```

e cioe' in modo confusionale e diverso dal voluto.
se ora battete RUN [RET]
si otterra' un

```
==>  ?SINTAX ERROR IN 5
==>  READY
```

togliete la linea 5 battendo:

5 [RET]

in pratica cio'significa sostituire la riga 5 con un' altra riga 5 ma con contenuto nullo.
poi battete RUN [RET]
facendo ripartire il programma si otterra' un:

```
==>  ?SINTAX ERROR IN 35
==>  READY
```

e quindi avete capito che non avete capito come si scrivono le linee di programma.
Provate ora a riscrivere le quattro istruzioni in modo corretto.

Per iniziare battete

NEW [RET]

questo comando serve per cancellare tutto quanto abbiamo immesso fino ad ora nalla RAM.

di seguito :

```
100 PRINT "QUESTA E' LA PRIMA PROVA"  [RET]
200 PRINT "SECONDA PROVA"             [RET]
358 PRINT "TERZA PROVA"               [RET]
5000 PRINT "QUINTA PROVA"             [RET]
10800 PRINT "QUARTA PROVA"            [RET]
```

e ancora battete :

[SHIFT] [CLR/HOME]

che serve per pulire il video, che e' l' unico a non avere memoria, da tutte le "sciocchezze" presenti.

LIST [RET]

riapparira' sul video (seguito da ==> READY col cursore lampeggiante) il vostro programma cosi' come l'avete scritto.

Provate ora a eseguire questo programma scrivendo

RUN [RET]

sul video appariranno le scritte contenute tra le virgolette. Il vostro COMMODORE 64 ha dunque eseguito le istruzioni che voi gli avete dato sotto forma di programma:

```
QUESTA E' LA PRIMA PROVA
SECONDA PROVA
TERZA PROVA
QUINTA PROVA
QUARTA PROVA

===> READY
```

Ora, per capire le capacita' del COMMODORE 64, muovete il cursore in modo da trasformare il 100 in 1000:
- posizionate il cursore all'inizio della riga 100 (come??? muovendosi coi tasti cursor. Se non vi e' chiaro rileggete il paragrafo che riguarda l'uso della

tastiera e state fermi un altro giro):

- battete 1000 [RET]
- battete [SHIFT] [CLR/HOME]
- battete LIST [RET]

ecco che nel vostro programma e' apparsa la riga 1000 esattamente uguale alla riga 100.
Se ora lanciate il programma (non dalla finestra... con la fatica che e' costata scriverlo, bensi' scrivendo RUN [RET]) constaterete che l'ordine delle prove non e' quello in cui si e' inserito il programma, ma quello stabilito dal numero della linea

- si puo' inserire una linea di programma assegnandole un numero interno ai numeri delle due linee tra le quali si intende inserire la nuova (esempio: per inserire una linea fra la 100 e la 200 potremo usare la linea 130; ecco perche' e' buona norma scrivere le istruzioni con numeri multipli di 10 in modo che l' inserimento di una riga sia sempre possibile).
- si puo' cambiare una linea assegnando lo stesso numero ad un'altra
- si puo' annullare una linea battendo il suo numero seguito da niente altro che [RET]

Provate sulla scorta di quanto detto a riordinare il programma in modo che battendo RUN si possa ottenere:

```
QUESTA E' LA PRIMA PROVA
SECONDA PROVA
TERZA PROVA
QUARTA PROVA
QUINTA PROVA

READY
```

Fate ora

LIST [RET]

ed apparira' il vostro programma, modificate il programma in modo da renderlo uguale al seguente:

```
1000 PRINT"QUESTA E' LA PRIMA PROVA"

1200 PRINT"SECONDA PROVA"
1300 PRINT"TERZA PROVA"
1400 PRINT"QUARTA PROVA"
1500 PRINT "QUINTA PROVA"
```

Cercate di eseguire i cambiamenti sforzandovi di utilizzare al massimo le linee esistenti senza dover eseguire una intera ribattitura (es. portate il cursore sulla linea 358 , battete 1300 sostituendo cosi' il vecchio numero di linea e [RET], poi spostate il cursore in basso fino alla prima linea completamente vuota e battete 358 seguito da [RET] ;questo serve per annullare la linea 358 che, altrimenti continua ad esistere, mentre non ha piu' senso.

Provate a battere RUN 1300 [RET] vi accorgerete che l'esecuzione parte dalla linea 1300 tralasciando le precedenti.

```
TERZA PROVA
QUARTA PROVA
QUINTA PROVA

READY
```

Diamo altre precisazioni sul comando LIST:

Questa istruzione serve per rivedere sul video tutte le istruzioni ordinate per numero di linea, e, quindi, per controllare cio' che abbiamo fatto; vediamone la sintassi e le possibilita':

```
provate a battere : LIST 1000      [RET]
                    LIST -1400     [RET]
                    LIST 1200-1400 [RET]
                    LIST 1200-     [RET]
```

potrete constatare che :

LIST 1000 provoca il listato dalla sola riga 1000
LIST -1400 lista tutto il programma fino a 1400
LIST 1200-1400 lista quanto c'e tra le due righe 1200 e 1400 comprese
LIST 1200- lista tutto il programma dalla riga 1200 alla fine

Per rallentare la velocita' del listato tenere premuto il tasto[CTRL] per arrestarlo battere [RUN/STOP].

Adesso respirate profondo e dedicate un po' di tempo a rileggere e ripensare a tutte le operazioni eseguite in queste ultime pagine!.

Un programma puo' essere arrestato con un comando diretto di stop che si ottiene premendo RUN-STOP oppure

inserendo l' istruzione STOP in una riga programma :

Battere

```
        1350 STOP  [RET]
        RUN        [RET]
```

noterete come il programma si arresti e segnali la linea di arresto.
Il programma puo' essere ripreso battendo direttamente CONT (iniziali di CONTINUE)

```
QUESTA E' LA PRIMA PROVA
SECONDA PROVA
TERZA PROVA


BREAK IN 1350    ( si e' fermato allo STOP)

READY


CONT  [ RET ]    ( gli diciamo di ripartire)

QUARTA PROVA
QUINTA PROVA

READY
```

Alla fine di un programma per arrestare l'elaborazione e' utile inserire l'istruzione END.

```
Battere 1350 END  [ RET ]
RUN               [ RET ]

QUESTA E' LA PRIMA PROVA
SECONDA PROVA
TERZA PROVA


READY              ( si e' fermato qui per il 1350 END )
CONT   [ RET ]  ( gli diciamo di ripartire   )

QUARTA PROVA
QUINTA PROVA
READY
```

Al READY, battere CONT ed osservare la differenza tra STOP ed END.(cioe' la mancanza nel secondo caso di BREAK IN 1350 cioe' della segnalazione di dove si e' fermato).

Prima di procedere ad analizzare tutti i comandi BASIC necessari per la programmazione, sara' bene tener presente che il COMMODORE 64 ammette anche una forma abbreviata dello stesso comando, cio' e' utilissimo in sede di battitura da tastiera di un programma, consentendo un notevole risparmio di tempo, il listato verra' comunque nella forma completa, cioe' se si batte una istruzione in modo abbrevviato, quando si esegue il LIST del programma essa ricompare per esteso.
In linea di massima la forma abbreviata si ottiene con la prima lettera del comando seguita dalla seconda battuta, pero', premendo contemporaneamente lo < SHIFT>

COMANDO	SI ABBREVIA CON	
END	E <SHIFT> N	eN
RUN	R <SHFIT> U	rU
CONT	C <SHIFT> O	cO
LIST	L <SHIFT> I	lI
STOP	S <SHIFT> T	sT

in caso di comandi con le prime due iniziali uguali si preme la terza contemporaneamente allo <SHIFT>:
fa eccezione

PRINT che si semplifica in ?

mentre P <SHIFT> R da PRINT# che avremo modo di esaminare piu' avanti.
Eventualmente per rendere piu' comprensibile quanto state scrivendo battete
[SHIFT] [C=] e passate al modo caratteri minuscoli / maiuscoli come spiegato nel paragrafo dedicato all' uso della tastiera, in questo modo vedrete il primo carattere del comando di tastiera in minuscolo ed il secondo in maiuscolo, anziche' il primo maiuscolo seguito dal simbolo grafico che leggete sulla destra del la parte anteriore del tasto.

Una istruzione necessaria per la gestione (o meglio distruzione) dei programmi e' NEW da utilizzare SEMPRE CON CAUTELA in quanto il suo uso come comando diretto provoca la perdita dell'intero programma (in realta' il calcolatore azzera solo delle variabili di riferimento detti puntatori, ma salvo esserre molto pratici una volta battuto NEW il programma E' PERSO)

Battete NEW [RET]
e successivamente LIST [RET]

Vedrete apparire solamente READY in quanto tutte le linee del programma precedente sono state cancellate.
Si puo' inserire NEW in un programma (diciamo alla fine al posto di END) in modo che alla fine lo stesso si autocancelli.
Potete anche inserire NEW nel programma di qualche vostro amico, e' sempre una cosa di effetto.
Esistono anche altri metodi per provocare un NEW ed in genere una sequenza di *@!??>"!@ da parte dell'operatore:.
Inciampare casualmente nel cavo di alimentazione transitando nei dintorni del COMMODORE 64 (di solito i parenti sono eccezionalmente dotati in questo sport)
 classico attimo di mancanza di corrente piu' o meno casuale (non accendere in casa piu' di 64 COMMODORE 64 contemporaneamente!!!).
Per essere cautelati nei confronti di tali eventi e' bene salvare spesso il programma sulle unita' periferiche (registratore o floppy) opportune anche se lo stesso non e' ancora finito.
Questa operazione va fatta evitando di cancellare l'ultima copia del programma (non registrate il nuovo sul vecchio) perche' il BLACK OUT in agguato potrebbe coglervi proprio in quell' istante e rovinare le due copie contemporaneamente

Un' ultima nota di carattere generale e' per l'istruzione REM (iniziale di REMARK = nota, notazione).
Si puo' definire un NON COMANDO in quanto il COMMODORE 64 ignorera' sistematicamente tutto cio' che appare a destra di REM fino alla fine riga.
Attenzione: la riga basic puo' essere scritta fino ad un massimo di 80 caratteri (di piu' se usate le istruzioni abbreviate, ma avrete problemi tutte le volte che vorrete modificarla) essa cioe' corrisponde a due righe schermo di cui la seconda viene attivata solo dopo aver riempito interamente la prima. Cio' avviene in quanto e' possibile separare piu' comandi col carattere ":"
Dopo il REM tuttavia i : vengono considerati solo segni di punteggiatura

Provate a battere:

```
NEW       [RET]

10 PRINT "REM":REM STAMPIAMO IL NOME REM
20 PRINT "REM REM REM":REM:STAMPIAMO REM REM REM
30 REM QUESTA RIGA NON SERVE AL C 64 MA SOLO A NOI
40 END
RUN     ( e perche' no rU )   [RET]
```

Ecco che il COMMODORE 64 ha eseguito il programma come se fosse stato scritto cosi'

```
10 PRINT "REM"
20 PRINT "REM REM REM"
40 END
```

Utilizzate spesso il REM per ricordarvi cosa state facendo, soprattutto se il programma e' complicato, cio' provochera' perdita di memeoria ma rendera' il programma piu' facilmente comprensibile a distanza di tempo.

LE VARIABILI NEL BASIC

Dal punto di vista matematico non avrebbe senso usare un calcolatore per eseguire delle operazioni esclusivamente tra numeri (che usualmente vengono detti costanti) anche se cio' e' possibile

```
PRINT 6.3+30-5*10  ( ricordate che si puo' abbre-
                     viare PRINT con ? e quindi
? 6.3+30-5*10

-13.7

READY
```

se non capite perche' proprio -13.7 non ha importanza.

Con questo metodo ogni volta che si cambia un numero (es. 30), bisogna modificare l'istruzione o il programma:
la cosa e' decisamente scomoda !.
Si ricorre allora all'uso di lettere che di volta in volta possono assumere valori diversi,
da qui il nome "VARIABILI"
Riprendiamo l'esempio e scriviamolo sotto forma di programma :

```
10 A=30
20 PRINT 6.3+A-5*10

RUN

-13.7

READY
```

modifichiamo la linea 10

```
10 A= 100

RUN

56.3

READY
```

Se ora con un'altra istruzione (INPUT A , che vedremo meglio nel prossimo paragrafo) riusciamo di volta in volta a passare i dati al COMMODORE 64 senza modificare il programma, il gioco e' fatto.

```
10 INPUT A

RUN


?
```

Ora il calcolatore si arresta VISUALIZZA UN ? nell'attesa che venga deciso il valore di A
provate a battere

200 [RET] (abbiamo assegnato alla variabile chiamata A il valore 200)

```
156.3

READY
```

Potete ora riprovare ad eseguire il programma dando diversi valori: la variabile A ogni volta si trasformera' nel valore assegnato e lo rappresentera' durante l' operazione, naturalmente potrete eseguire operazioni molto piu' complicate ed utilizzare i diversi tipi di variabili esistenti.

Le variabili previste dal BASIC sono di tre tipi:

- variabili numeriche intere
- variabili numeriche reali
- variabili alfanumeriche

VARIABILI NUMERICHE INTERE

Vengono rappresentate con sigle di una, due o piu' lettere di cui

-la prima sempre e solo alfabetica
-seguita da un'alfabetica o numerica
-e dal simbolo %.

Per comodita' e chiarezza le variabili possono avere piu' di due lettere (ultima sempre la %) ma il COMMODORE 64 riconosce solo le prime due e il %, percio' due variabili diverse devono differire tra loro nelle due lettere iniziali oppure saranno confuse come fossero la stessa.

VARIABILI VALIDE	VARIABILI ERRATE
AA	1A%
Z1%	A%X
LXY%	A!%
VARIABILE%	*A%
V%	R>%

-Il contenuto assegnato ad una variabile intera deve essere compreso tra:

-32768 fino a +32767

-nel caso si usasse come variabile per rappresentare un numero con dei decimali
il COMMODORE 64 ne prendera' la sola parte intera.
provate i seguenti tre programmi:

```
NEW
10 VA%=9
20 PRINT VA%
30 VARIABILE%=18
40 PRINT VA%
```

```
NEW
10 VA%=5.32
20 PRINT VA%
```

```
NEW
10 VA%=65000
20 PRINT VA%
```

Cercate di prevedere prima cosa avverra' battendo RUN per i tre programmi.
Per conoscenza diciamo che una variabile intera nella memoria del vostro COMMODORE 64 occupa 2 bytes ossia 2 caselle di memoria.

VARIABILI NUMERICHE REALI

A differenza delle variabili intere, queste sono in grado di rappresentare qualsiasi numero da:

-2.93873588 E-39 fino a 1.70141183 E+38

naturalmente anche con i decimali. Nella variabile reale cio'avviene in forma normale ed in forma esponenziale a seconda del valore che assegnamo alla nostra variabile.
La precisione sul numero e' di 10 cifre, di cui 9 vengono visualizzate arrotondando al meglio anche la decima.
Tenete presente che E-39 significa dividere -2.93... per un numero fatto da 1 seguito da 39 zeri (la trentanovesima potenza di dieci) mentre E+38 significa 1.70.... moltiplicato per la trentottesima potenza di 10

100000000000000000000000000000000000000 per 1.70141183

e se gli zeri non sono 38 vuol dire che abbiamo sbagliato a stamparli

Provate: NEW

```
10 A=10.2747879407
20 PRINT A
```

RUN

10.2747879

avrete perso 407 anche se 4 e' in memoria e 07 no

modificate A= 10.27478797

RUN
10.274788

797 e' stato arrotondato a 800

ancora A= 102747879707

RUN

1.0274788 E+11
(ossia 1.02747879 per 100000000000)

ancora A= 0.00000001027478797

RUN

```
1.0274788 E-09
```

naturalmente anche noi possiamo assegnare al COMMODORE 64 i dati nello stesso modo

```
10 A= 10.2747879707 E-10
20 PRINT A
RUN
1.0274788 E-09
```

-per indicare una variabile reale si usa lo stesso criterio gia' esposto per le variabili intere, ma si omette il simbolo % :

VR e' una variabile reale dove V e' sempre alfabetica R puo' essere anche un numero

VARIABILI ESATTE	VARIABILI ERRATE
V7	V!
VARIAB	1E
V1X	*D

per il vostro calcolatore 2 variabili di uguale nome, una reale ed una intera sono cose diverse e quindi non confondibili.
Provare:

```
NEW
10 AZ% = 5
20 AZ  = 26.30
30 PRINT AZ%
40 PRINT AZ
50 PRINT AZ% * AZ :REM PRIMO PRODOTTO
60 PRINT AZ% / AZ :REM QUOZIENTE
RUN
```

Come potrete constatare, dopo l' esecuzione di questo programma,il COMMODORE 64 distingue le due variabili anche se portano lo stesso nome.
Battete ora NEW ed il seguente programma:

```
10 A1 = 15E+15
20 A2 = 2E-15
30 PRINT A1/A2
40 PRINT A2/A1
```

Potrete rendervi conto cosi' che tutte le operazioni lasciano residui (detti errori di troncamento) per cui

il risultato presenta un lieve errore sull' ultima cifra.

Poiche' un numero intero e'un particolare tipo di numero reale, si puo' sempre definire una variabile reale uguale ad una intera:

V1=A%   e' esatto

non e' pero' vero il contrario, infatti assegnando ad una variabile intera il valore di una reale si perderebbero tutti i decimali, come nel seguente esempio (da provare dopo il solito NEW):

```
10 V1 = 100.36
20 A%=V1
30 PRINT A%
```

Se l'utilizzo di variabili intere puo` provocare dei fastidi viene spontaneo domandarsi quale utilita` puo` esservi ad usarle in luogo delle corrispondenti reali.
- in primo luogo l'occupazione di memoria per un reale e` due volte e mezzo superiore a quello necessario per memorizzare un intero (una variabile reale occupa infatti ben 5 bytes).
- in secondo luogo esistono dei casi in cui e' necessario usare delle variabili come contatori aventi precisione assoluta per evitare che il confronto di due numeri possa poi portare ad errori.

VARIABILI ALFANUMERICHE

Possono contenere lettere dell' alfabeto, simboli o numeri e sono comunemente chiamate stringhe.
Il COMMODORE 64 ha la possibilita' di utilizzare stringhe, cioe' variabili alfanumeriche formate con qualsiasi carattere alfabetico, numerico, grafico segnato in tastiera, fatta eccezione per le " usate per definire la stringa stessa. La lunghezza massima consentita e' di 255 caratteri.
Una variabile alfanumerica viene rappresentata con le solite due lettere (di cui la prima solo alfabetica e la seconda anche numerica) seguite dal $.

Variabili OK	Variabili errate
AB$	A<$
C1$	A"$
STRINGA$	1A$

Provate

```
NEW
10 A$="A$ E' UNA STRINGA"
20 PRINT A$

RUN

A$ E' UNA STRINGA
```

Battete ora NEW

```
10 A$="1520+1000"
20 PRINT A$

RUN

1520+1000
READY
```

Se vi aspettavate il risultato della somma cioe' 2520 dovete ancora meditare sul significato di variabile alfanumerica, infatti tutto quanto e' compreso tra "" viene usato come una unica parola senza alcun significato numerico.
Quindi la funzione delle variabili alfanumeriche (o stringhe) e' quella di contenere qualsiasi carattere anche non numerico. L' occupazione di memoria di una variabile stringa e' pari a due bytes + il numero di caratteri contenuti nella stringa, cioe' al numero di caratteri assegnati alla variabile. Capirete meglio l'

uso e la funzione delle stringhe continuando nella lettura di questo libro.

DEFINIZIONE DELLE VARIABILI SECONDO LA LOGICA DEL COMMODORE 64

Per mettere in condizione il nostro cervellone di funzionare sempre bene bisogna assegnare il valore di un dato (o altra variabile) gia' noto al computer perche' definito in precedenza; la forma e':

```
LET A1=B6     LET A1=3.14
```

dove A1 deve essere ancora definito e B6 e' gia' noto. Nel caso cio' non avvenga (cioe' B6 non sia stato ancora definito) il COMMODORE 64 pone le variabili a valore nullo.

DEFINIZIONI OK	DEFINIZIONI ERRATE
LET A1=15,73	LET A1="STRINGA"
LET A2%=18	LET A2$=19,36+B2$
LET A3$="STRINGA"	LET A3$=28,33
LET A3$="A2%"	LET A3$=A2%
A1= 15.73	5=A1
LET A3$="V=15,26"	"ERRORE"=A4$
	A$+B$="SOMMA"

Si consideri che LET puo' essere anche tralasciato perche' se il vostro COMMODORE 64 trova A = 1 sottintende "poni A = 1" (LET in inglese vuol dire poni).
Provate i seguenti programmi:

```
NEW
5 PRINT A1
10 A1=15,73
20 PRINT A1
```

battete ora RUN per eseguirlo

```
NEW

5 PRINT A1
10 A1=10
15 A1=A1+1
20 PRINT A1
```

RUN per l' esecuzione e successivamente NEW per il prossimo programmino

```
5 PRINT A1
10 A1=10
20 A1=A1+A1
30 PRINT A1
```

RUN

Il COMMODORE 64, come avrete notato dagli esempi, accetta di ridefinire una variabile usando se' stessa:

A1=A1+A1 significa:
somma il contenuto della variabile A1 a se stesso e mettilo nella variabile A1 cioe':

```
 A1 = 10
 A1 = 20 = 10 + 10
 A1 = 20
```

Si poteva anche scrivere:

```
20 B1= A1+A1
21 A1= B1
```

ma si sarebbe sprecata memoria e tempo, e con una forma abbastanza scorretta. Proviamo ora a lavorare un po' con le variabili alfanumeriche:
battete questo programmino:

NEW

```
10 A$ = "SOMMA "
20 B$ = "DI STRINGHE"
30 C$ = A$+B$
40 PRINT C$
50 PRINT C$ + 5
```

RUN

Notate che le virgolette della riga 10 sono state chiuse uno spazio dopo la lettera A; questo spazio e' stato ottenuto da tastiera premendo la barra spaziatrice ed e' necessario per evitare di ottenere una variabile C$ contenente la scritta "SOMMADI STRINGHE" che sarebbe sintatticamente errata. L' operazione di somma tra le due stringhe A$ e B$ ha provocato la "concatenazione" dei contenuti delle due variabili che sono stati disposti l' uno in seguito all' altro senza

interposizione di spazi. Poiche' esiste un errore in riga 50 provate ad eliminarlo in modo che appaia:

```
SOMMA DI STRINGHE + 5
```

(dovete incapsulare 5 tra le virgolette facendolo diventare la stringa "5").

Una stringa puo' essere composta anche di soli spazi bianchi (indicati nella programmazione con una b tagliata da una riga diagonale, questo perche' e' piuttosto difficile riconoscere uno spazio bianco su di un foglio bianco se non e' del tipo a quadretti).

```
NEW

10 A$ ="          ":REM 10 SPAZI BIANCHI
```

dove tra le virgolette e' stata battuta la barra spaziatrice 10 volte.

```
20 PRINT A$
RUN
```

Non viene evidenziato nulla, tuttavia e' possibile ottenere un'idea chiara di cosa succede inserendo le seguenti linee di programma:

```
15 A$ ="< CTRL > 9"+A$:REM VISUALIZZA I CARATTERI SUCCESSIVI IN REVERSE
```

ora il RUN evidenzia la variabile A$ in reverse sullo schermo (il reverse di un carattere vuoto e' un carattere tutto pieno);
si puo' fare di piu':

```
16 A$ ="< CTRL > 8"+A$:REM MODIFICA IL COLORE AI CARATTERI SEGUENTI
```

e la striscia si sara' colorata in giallo.
Potete ora provare a far apparire piu' colori con lo stesso criterio, inserite la linea:

```
17 A$=A$+" < CTRL > 3" + A$ + " < CTRL > 7" + A$
```

e di nuovo RUN

Si puo' notare che all'interno delle virgolette determinati tasti non creano piu' lettere ma comandi per la modifica del video e cio' e' utilissimo per una miglior composizione dei vostri programmi, cio' verra' analizzato in esteso nel successivo paragrafo (PRINT).

OPERATORI DA NON USARE

Non usate mai come variabili i seguenti nomi: TI, TI$, ST perche' sono funzioni BASIC; inoltre non assegnate mai alle variabili i nomi delle altre funzioni BASIC (comunemente questa restrizione viene indicata come 'parole riservate'); quindi non potrete usare come nome di variabili nessuno dei comandi Basic:

- potrete avere una variabile RE ma non una variabile REM
- potrete avere una variabile DAT, che verra' usata come DA, ma non una variabile DATA

OPERATORI ARITMETICI TRA VARIABILI

Esaminiamo i seguenti caratteri che abbiamo in parte gia' visto:

+ provoca l'addizione di due variabili
- provoca la sottrazione di due variabili
* provoca il prodotto di due variabili
/ provoca il quoziente: la variabile di sinistra
  divisa per quella di destra
↑ provoca l'elevamento a potenza
(apre una parentesi
) chiude una parentesi

E' il caso di ricordare che il quoziente e l'elevamento a potenza tra numeri interi e' da usarsi con cautela. Infatti se il risultato di un quoziente viene assegnato ad una variabile intera si perdera' la parte decimale.

```
NEW
10 A% = 100/3
20 PRINT A%
RUN

33

READY
```

il risultato giusto si ha eliminando A% ed usando la variabile reale A.

Se superate con A% il valore di +32767, si ottiene segnalazione di errore.

```
NEW
10 A% = 100↑1000
20 PRINT A%
RUN

?OWERFLOW ERROR IN 10

READY

NEW
10 A% = 100↑(-10)
20 PRINT A%
RUN

0

READY
```

Anche in tal caso il risultato e' errato, ma non si ha segnalazione di errore.
L'unico operatore utilizzabile per le stringhe e' il +, che le lega ponendole una di seguito all'altra senza spazi intermedi anche perche' qualsiasi altro tipo di operazione sarebbe inpensabile es: quanto fa Fulvio * Fernando...? scriveteci se riuscite a risolverlo!.
Per ottenere altre modifiche sulle stringhe esistono altre istruzioni apposite che verranno descritte piu' avanti dettagliatamente.

Provate in modo diretto:

```
PRINT 100/3
33.3333333
READY

PRINT 10↑3
1000
READY

PRINT 10*10*10
1000
READY
```

Ed ora provate in modo programma:

```
NEW
10 A = 5+6*7
20 PRINT A
RUN

47
READY
```

Il COMMODORE 64 ha eseguito il calcolo come se fosse stato impostato cosi': 5+(6*7) perche' e' stata rispettata una PRECEDENZA (attenzione agli incroci, semafori tram ed altro), il prodotto tra due numeri viene sempre eseguito prima dell' addizione.
Il quadro della precedenze e' il seguente:

() operazioni tra parentesi
↑ elevamenti a potenza
*/ prodotto, quoziente
+- somma e differenza

Tra due segni ad ugual precedenza il calcolo viene eseguito nell'ordine da sinistra a destra.
Cercate di prevedere il risultato delle seguenti espressioni, poi inseritele e controllatene l'esattezza:

```
5 + 6 / 2 - 4 * 2
6 * 3 / 2 + 5 / 10
2 * 10 ↑ 2 / 4
```

Ora provate a scrivere l'espressione corrispondente al seguente calcolo:

A + 1/2 * (B - C)

B e C devono essere sottratti tra loro prima che si operi la divisione, avendo la differenza una precedenza piu' bassa del quoziente l'espressione seguente

A+B-C/2
sarebbe errata

A+(B-C)/2
e' invece esatta in quanto le parentesi hanno precedenza sul quoziente.

Se l'espressione fosse:

$$\left[A + \frac{B-C}{2}\right] \times (C+8)^3$$

si scrivera':

```
(A+(B-C)/2)*(C+8)↑3
```

che verra' risolta secondo il seguente schema:

```
(A + (B - C) / 2) * (C + 8) ↑ 3
(A +    D    / 2) *    E    ↑ 3    con D = B-C    E = C+8
(A +       F    ) *       G        con F = D/2    G = E↑3
        H         *       G
```

Maggior attenzione dovra' essere prestata quando si introducono altre funzioni ad esempio

$$B = SIN\ \frac{\pi Y}{L} + \sqrt{\frac{\pi Y}{L}}$$

va indicato con

```
B = SIN ( Π*Y/L) + SQR (Π *Y/L)
```

Altre volte potra' essere necessario semplificare l'espressione matematica in modo da rendere la programmazione piu' chiara:

```
A = Π * Y / L
B = SIN (A) + SQR (A)
```

altri esempi di trasformazioni:

RELAZIONI	OK	Errate o inutili
$A = 10^{1/2}$	A=10↑(1/2)	A = 10 ↑1/2
$A = \sqrt{10^3+7^5}$	B= 10↑3+7↑5 A=SQR (B)	A = SQR (10↑3)+(7↑5) A = SQR(10↑3. +(7↑5)
$A = \frac{B^{\frac{1}{C}} + B^{-\frac{1}{C}}}{2}$	D = 1/C E1= B↑D : E2 =B↑(-D) A = (E1+E2)/2 oppure A = (B↑(1/C)+ B↑((-1)/C)))/2	E1= B↑1/C E2= B↑ 1/C A = E1 + E2/2 oppure A=(B↑(1/C)+B↑(-1/C))/2

L'ULTIMA ESPRESSIONE E' ERRATA in quanto un numero dichiarato negativo deve venire compreso tra parentesi.

OPERATORI LOGICI TRA VARIABILI

Sono in grado di mettere in relazione due variabili e confrontarne il valore:

A = B	uguaglianza
A >< B	A diverso da B
A < B	A minore di B (o B maggiore di A)
A > B	A maggiore di B (o B minore di A)

Il minore dei due e' sempre quello contro cui e' puntato il vertice del simbolo grafico. I simboli possono essere ulteriormente combinati:

A < = B	A minore od uguale a B
A > = B	A maggiore od uguale a B.

Ha senso anche un confronto tra stringhe:

A$ = "CONFRONTO TRA STRINGHE"
B$ = A$ sara' vero se anche in B$ esiste una stringa con

la stessa sequenza di caratteri di A$, compresi anche gli spazi bianchi.

A$ = "F"
B$ = "H"
in tal caso risultera' A$ < B$

in quanto il carattere H occupa una posizione piu' elevata di F nella tabella caratteri, che si esaminera' con l'istruzione CHR$ (vedere anche la Tabella riportata in Appendice)

Con lo stesso criterio:

"ALFANUMERICO" e' maggiore di "ALFANUMERICA" in quanto le due stringhe sono uguali in tutto fuorche' nell'ultima lettera che diventa cosi' criterio di confronto.

"ALFABETA" e' minore di "ALFAGAMMA" in quanto "B" e' < di "G", mentre "ETA", "AMMA" e tutto il resto delle stringhe dopo il rilevamento della disuguaglianza non viene piu' considerato.
Infine:

"ALFA" e' minore di "ALFABETA"

perche' la seconda stringa e' piu' lunga della prima.

Prima di passare allo studio delle successive istruzioni basic si provino i seguenti programmi:

```
10 A=2 : B=100 : PRINT " < CTRL > 9 RADICI QUADRATE"
15 PRINT" SUCCESSIVE DI 100"
20 B=B ↑ (1/A) : PRINT B
30 B=B ↑ (1/A) : PRINT B
40 B=B ↑ (1/A) : PRINT B
50 B=B ↑ (1/A) : PRINT B
60 B=B ↑ (1/A) : PRINT B
70 B=B ↑ (1/A) : PRINT B
80 B=B ↑ (1/A) : PRINT B
```

Dalla RIGA 30 fino a RIGA 80 ribattere la stessa istruzione semplicemente alzando il cursore e modificando il numero di riga (eseguire (SHIFT) CRSR 4 (RETURN) etc.

```
90 PRINT " < CTRL > 9 ORA RILEVIAMO AL QUADRATO"
100 B=B↑A : PRINT B
110 B=B↑A : PRINT B
120 B=B↑A : PRINT B
130 B=B↑A : PRINT B
140 B=B↑A : PRINT B
150 B=B↑A : PRINT B
160 B=B↑A : PRINT B
170 B=B↑A : PRINT B
```

dalla RIGA 110 alla RIGA 170 usare lo stesso metodo sopradescritto.

```
180 PRINT "CTRL 9 IL RISULTATO FINALE E' CAMBIATO"
190 PRINT "CTRL 9 INFATTI VALE CRTL 0" B
```

e cio' a causa degli arrotondamenti successivamente eseguiti in ogni operazione di estrazione della radice quadrata di B

```
NEW

100 REM CALCOLO DEL VOLUME DI UNA SFERA
110 REM DI RAGGIO 100 CON LA FORMULA 4/3 π R↑3
120 R=100
130 V=4/3 * π * R↑3
140 PRINT " < CTRL > 9 IL VOLUME E' DI < CTRL > 0" V
150 END
```

Il simbolo π si ottiene premendo <SHIFT> ↑

```
NEW

80 REM QUESTO PROGRAMMA E' MOLTO COLORATO
90 A$ = "                    ":REM 20 SPAZI BIANCHI
100 A$ = " < CTRL > 9" + A$   :REM ATTIVA IL REVERSE
110 B1$ = "< CTRL > 1":B2$="< CTRL > 2":B3$="< CTRL > 3"
120 B4$ = "< CTRL > 4":B5$="< CTRL > 5":B6$="< CTRL > 6"
130 B7$ = "< CTRL > 7":B8$="< CTRL > 8
140 C1$ = "< C= > 1 ":C4$ ="< C= > 2 ":C3$="< C= > 3"
150 C4$ = "< C= > 4 ":C5$ ="< C= > 5 ":C6$="< C= > 6"
160 C5$ = "< C= > 7 ":C8$ ="< C= > 8 "
200 PRINT B1$ + A$
210 PRINT C4$ + A$
```

Ora usate le linee 200 e 210 per generare le successive :ad esempio dopo aver battuto la riga 200 e 210 riportate il cursore su 200 e correggete in 220 <RET>

correggete 210 in 230 <RET> ; ora: <SHIFT> <CLR/HOME> LIST <RET>; riportate il cursore su 200 e trasformatelo in 240 <RET>, il 210 in 250 , il 220 in 260, il 230 in 270 (sempre convalidandoli con <RET>. Ora riportate il cursore su 270 e correggetelo in 280 <RET> poi ancora in 290 <RET> ; fatto questo pulite il quadro e ridate il <LIST>.
Col cursore modificate 200 in 300 e cosi' via fino a 350.

Date un ultimo <LIST> ed ora preoccupatevi di correggere B1$ nel carattere corrispondente ad ogni riga (potete anche evitare di convalidare ogni volta la riga, eviterete di riportare ogni volta il cursore in posizione, ma dovrete poi alla fine convalidare tutte le righe cominciando dalla piu' alta e scendendo sempre col solito tasto < RET>).

```
200 PRINT B1$ + A$
210 PRINT C44 + A$
220 PRINT C5$ + A$
230 PRINT C8$ + A$
240 PRINT B2$ + A$
250 PRINT B8$ + A$
260 PRINT C1$ + A$
270 PRINT C3$ + A$
280 PRINT B5$ + A$
290 PRINT C2$ + A$
300 PRINT B3$ + A$
310 PRINT B7$ + A$
310 PRINT C7$ + A$
320 PRINT B4$ + A$
330 PRINT C6$ + A$
340 PRINT B6$ + A$
350 GOTO 200
```

Finalmente se tutto va bene date il RUN. Alla fine farete fatica a leggere a causa dei colori, percio' eseguite in modo diretto < CTRL > 1 o < RUN/STOP > < RESTORE >. Provate anche ad aggiungere la riga :

```
95 A$ = A$ + A$ + A$
```

Subito dopo aver convalidato RUN col <RET> tenete premuto CTRL e il video rallentera' la sua carellata di colori.
Utilizziamo queste caratteristiche per eseguire un elementarissimo grafico necessario per rappresentare

l'andamento di un dato nei 12 mesi.
Modificate le seguenti righe:

```
90 B$ = "  ":C4 = "    ":D$ = "      ":E$ = D$ + D$
95 A$ = " < CTRL > 9" + "  "
```

Eliminate le righe 100, 210,220,230 e correggete la 240

```
90 B$ = "  ":C$ = "    ":D$ = "      ":E$ = D$ + D$
95 A$ = "<CTRL>9"+"  "
240 PRINT B1$+A$+"GENN."
250 PRINT B8$+A$+B$+"FEBBR."
260 PRINT C1$+A$+B$+"MARZO"
270 PRINT C3$+A$+D$+"APR."
280 PRINT B5$+A$+D$+B$+"MAGG."
290 PRINT C2$+A$+D$+C$+"GIU."
300 PRINT B3$+A$+E$+"LUGL."
310 PRINT 37$+A$+E$+B$+"AGO"
320 PRINT C7$+A$+E$+C$+"SETT."
330 PRINT B4$+A$+E$+D$+"OTT."
340 PRINT C6$+A$+E$+D$+B$+"NOV."
350 PRINT B6$+A$+E$+D$+B4+C$+"DIC."
```

Successivamente vedremo anche come sia molto piu' facile ottenere strisce di lunghezza desiderate e di colore desiderato senza dover ricorrere a programmi cosi' lunghi.

ISTRUZIONI PER RICAVARE O COMUNICARE DATI AL PROGRAMMA

PRINT

E' un'istruzione gia' vista in precedenza e serve per stampare; vediamone ulteriori particolarita':

- Piu' variabili devono essere separate tra loro da un segno di interpunzione:
(li riportiamo di seguito tra virgolette):

";" le variabili vengono stampate una di seguito all'altra;

"," le variabili vengono stampate a partire dalla colonna multiplo di 10 successiva all'ultima stampa. (se l'ultimo carattere stampato e' in colonna 16 la stampa del numero successivo parte da colonna 20).

Se volete che il successivo PRINT continui la stampa dal punto in cui si e' arrestata la precedente, l'ultima variabile dell'ultimo PRINT deve essere seguita dall'opportuno carattere:

```
10 PRINT 5,10,3.27,"CON LA VIRGOLA"
20 PRINT 5;10;3.27;"COL PUNTO E VIRGOLA"
30 PRINT 5.55,7.2777,111111.02,666.666,9999.2
40 A=50.7:PRINT "LA VARIABILE A E' =";A
```

notate che:

- ogni stampa e' partita dall'inizio della riga successiva. I numeri della riga 30 non sono incolonnati rispetto al punto decimale, ma tutti sul lato sinistro.
- nonostante non si siano lasciati spazi bianchi dopo l'uguale (riga 40) il numero 50.7 appare spostato di uno spazio necessario per il segno (che essendo positivo non appare).

```
5000 A=-50.7:PRINT "LA VARIABILE A E' A=";A

RUN
```

(noterete che sull' ultima riga appare il numero con il segno)

battete ora NEW:

```
5 A=10:B=20:C=30
10 PRINT "TRE NUMERI", A;
20 PRINT B
30 PRINT C
40 PRINT "FINE DATI"

RUN
```

Il 30 e' stato stampato a capo in quanto manca il punto e virgola dopo B.
Correggere e riprovare.

- E' possibile provocare lo spostamento del cursore usando delle particolari funzioni all'interno delle stringhe, queste funzioni si attivano solamente quando e' stato premuto per la prima volta il simbolo " (finalmente chiariamo il perche' di tanti simboli strani che fanno spostare il cursore durante l' esecuzione

dell' istruzione PRINT come se voi aveste premuto i tasti di controllo cursore).

```
20 PRINT"▯":PRINT"20";:REM < SHIFT >< CLR/HOME>
30 PRINT"▮▮▮";:PRINT"30";:REM < CRSV >
40 PRINT"▮▮▮";:PRINT"40";:REM <SHIFT> < CRSH >
50 PRINT"▯▯▯";:PRINT"50";:REM <SHIFT> < CRSV >
```

Analizziamo il programma per capire:

- in riga 20 la prima istruzione pulisce lo schermo, la seconda stampa sul video 20 ed avverte il cursore di non andare a capo, perche' avendo messo il ";" , la stampa successiva deve continuare da quel punto

- in riga 30 il cursore si abbassa di 3 linee, stampa 30 e passa alla successiva sempre senza spostare il cursore (con il ";")

- in riga 40 il cursore avanwa di tre colonne, stampa 40
- in riga 50 ,partendo dall' ultimo punto (sempre per ";") si alza di 3 righe e stampa 50

Se poi volete ottenere i 4 numeri sui vertici di un rettangolo, inserite le seguenti due linee:

```
25 PRINT"▮▮▮▮";:REM < SHIFT > <  CRSRH >
45 PRINT"▮▮▮▮";:REM < SHIFT > <  CRSRH >
```

Questi comandi provocano l' arretramento del cursore di tre spazi dopo la stampa del numero.

Altri spostamenti del cursore si possono assegnare con le funzioni:

TAB (n) oppure SPC (n).

NEW

```
10 PRINT "PRIMA STAMPA"TAB(20)"TABULATORE"
20 PRINT SPC(15)"SPOSTAMENTO"
30 PRINT SPC(15)"SPOSTAMENTO";TAB(20)"TABULATORE"
```

La differenza tra i due operatori e' la presente:

- TAB(20) sposta in avanti la stampa per farla avvenire con inizio in colonna 20 (cio' indipendentemente dalle stampe precedenti salvo che queste non superino la colonna il cui numero e' inserito nel TAB.).
- SPC(15) sposta la stampa di 15 colonne dall'ultimo punto in cui e' avvenuta la stampa precedente.
Naturalmente in riga 30 TAB 20 non puo' funzionare perche' si e' giunti a colonna 26 e quindi la stampa prosegue dall'ultima colonna stampata.
Queste funzioni, unite a quelle per il trattamento delle stringhe, vi permetteranno di poter ottenere tabulati etc.

INPUT

Questa istruzione blocca l'elaborazione, visualizza un ? e attende il valore del dato che deve essere inserito nel programma da tastiera quale valore della variabile o delle variabili specificate nell' istruzione input.
L'operatore deve digitare il dato da tastiera e premere <RETURN> assegnando cosi' il valore desiderato alla variabile prevista in quel punto dal programma.

La sintassi di INPUT e' la seguente:

INPUT "variabile A";A,B,C,.....

dove la scritta "variabile A"; e' opzionale, cioe' puo' essere tralasciato scrivendo semplicemente:

INPUT A,B,C,....

Tuttavia nel caso si dovessero inserire numerosi dati, operando nell'ultimo modo, sara' ben difficile ricordare a che punto si e' giunti.

```
10 PRINT"■CALCOLO DELL' AREA DI UN TRIANGOLO"
20 PRINT"■INSERIRE LA BASE";:INPUTB
30 INPUT"■ALTEZZA";H
40 PRINT"■■■L' AREA E' ";B*H/2
```

appare

```
CALCOLO DELL' AREA DI UN TRIANGOLO

INSERIRE LA BASE ?
```

a questo punto digitare 400 <RETURN>
appare:

```
CALCOLO DELL' AREA DI UN TRIANGOLO

INSERIRE LA BASE ? 400

ALTEZZA ?
```

digitare ad esempio 500 <RETURN>
apparira':

```
CALCOLO DELL' AREA DI UN TRIANGOLO

INSERIRE LA BASE ? 400

ALTEZZA ? 500


L' AREA E'  100000
```

Notate come nel programma precedente si siano combinate le istruzioni PRINT ed INPUT per informare sul video l' operatore in merito a quali siano i dati richiesti in quell' istante dal programma.
Ripetere il RUN, ed alla prima richiesta battere semplicemente <RETURN> senza alcun dato; il COMMODORE 64 assegnera' valore nullo alla variabile richiesta e proseguira' nell'elaborazione. Questo puo' anche essere un inconveniente perche' se inavvertitamente il tasto <RETURN> viene premuto piu' di una volta si perde la possibilita' di inserire il dato nella posizione desiderata.
Cio' avviene perche' i caratteri assegnati da tastiera vengono "parcheggiati" in un'area (detto Buffer di tastiera) nell'attesa di essere prelevati dal programma (si possono battere sino a 10 caratteri); a volte al ? creato dall'INPUT seguono quei caratteri che avevate inavvertitamente premuto prima che l' elaborazione giungesse all' INPUT stesso. Dovrete semplicemente annullarli per mezzo dei soliti tasti di controllo cursore

INST/DEL

Con una istruzione INPUT si possono assegnare piu' dati, basta semplicemente separare le variabili da definire con una ","

```
10 PRINT"CALCOLO DELL'AREA DI UN TRAPEZIO"
20 INPUT"BASE MAGGIORE  BASE MINORE";B1,B2
25 B=(B1+B2)/2
```

Alla domanda di riga 20 battete ad es. 50,250 [RET] anche i dati sono stati assegnati divisi dalla ","

Battere NEW ed iserire il seguente programma:

```
10 INPUT"NOME E COGNOME";A$
20 INPUT"LUOGO DI NASCITA";B$
30 INPUT"ANNO DI NASCITA 19?";N%
40 PRINT" IL SIGNOR ";A$
50 PRINT" E' NATO A ";B$
60 PRINT" BEN ";1983-N%;" FA"
```

Alle richieste del programma potrete assegnare un qualsiasi nome (il signor A27BCD e' nato a BOS27><) ma alla terza richiesta non si potra' assegnare un dato alfanumerico in quanto il COMMODORE 64 richiedera' il dato nuovamente con:

REDO FROM START.?

quando invece si assegna un dato reale (esempio 1953,72) ad una variabile intera, il COMMODORE 64 tralascera' i decimali (ponendo A%=1953).

GET

E' similare all' istruzione INPUT salvo che legge dalla tastiera (anzi dal Buffer di tastiera) un solo carattere per volta senza bisogno del tasto <RET> per far proseguire il programma; questa istruzione tuttavia potra' essere correttamente usata solo dopo che avrete preso in esame altri comandi (IF THEN, GOTO etc.) percio' per ora "pazientare".

DATA & READ

Esistono dei dati che non vale la pena di inserire da tastiera e richiederebbero troppo spreco di memoria se definiti all'interno del programma usando la classica istruzione LET, in tal caso si utilizza l'espressione READ per ordinare al calcolatore di leggere il valore delle variabili e DATA per memorizzare questi valori nel programma stesso.

```
10 READA$,B$,C$,D$,E$
20 PRINTA$:PRINTB$:PRINT"▩"C$:REM < CRSR >
21 PRINTTAB(3)D$TAB(17)E$:PRINT:PRINTC$:PRINT
22 READA$,A:PRINTTAB(3)A$TAB(17)A
23 GOTO22
30 DATA"SALVE QUASTA E' UNA PROVA"
40 DATA "PER CAPIRE COME FUNZIONA READ-DATA"
50 DATA"*--------------*---------*"
60 DATA "MESE","N.GIORNI"
70 DATA"GENNAIO",31,"FEBRAIO",28,"MARZO",31
80 DATA"APRILE",30,"MAGGIO",31,"GIUGNO",30
90 DATA"LUGLIO",31,"AGOSTO",31,"SETTEMBRE",30
100 DATA"OTTOBRE",31,"NOVEMBRE",30,"DICEMBRE",31
```

Il programma contiene un'istruzione (22 GOTO 21) non ancora nota: e' l'ordine di tornare ad eseguire la riga 21 (e cio' avverra' indefinitamente fino a che il COMMODORE 64, avendo esaurito i dati da leggere segnalera' un OUT OF DATA ERROR, cioe' ho finito i dati!):

Il calcolatore in RIGA 10 legge i primi 4 dati (non importa se questi sono lontani in riga 30,40...) e li usa per definire le variabili A$,B$,C$,D$.

Giunto alla riga 21 il COMMODORE 64 legge A$,A (ossia GENNAIO,31) lo stampa e poi con riga 22 ritorna a definire le stesse variabili A$,A con i dati successivi (FEBBRAIO,28) etc.

Senza questo metodo si sarebbero dovute utilizzare 24 variabili diverse e scrivere 12 comandi di PRINT.

Questa procedura si potra' ancora sensibilmente sveltire con l'utilizzo dei "vettori" ossia di variabili con indice.

Naturalmente bisognera' fare attenzione di far corrispondere il tipo di variabile dell' istruzione READ al contenuto della istruzione DATA, infatti non potrete ordinare la lettura di un reale ed inserire poi una stringa come dato (se fosse il contrario, cioe' READ

A$....DATA 5.27 il programma funzionera' ?) infatti si avrebbe segnalazione ?SINTAX ERROR IN ... seguito dal

numero della riga DATA contenente il dato non corrispondente (e fate attenzione che non e' facile accorgersene).

Il COMMODORE 64 tiene il conto dei dati letti e da leggere con un contatore che punta nella lista il dato successivo a quello appena letto in modo da consentirne il rapido prelevamento all'interno del programma in esecuzione. Cio' e' indipendente dalla posizione del dato all'interno della lista delle istruzioni, conta solo:

- la sua posizione nei confronti dei dati precedenti e successivi, sia sulla riga dati che rispetto alle altre righe dati.
- la sua corrispondenza con l'istruzione READ che lo deve prelevare;
- il numero dei dati da leggere deve essere uguale o maggiore al numero di volte in cui viene eseguita l'istruzione READ;
- la variabile da leggere deve corrispondere al tipo di dato. (Reale con reale, intero con intero, variabile alfanumerica con stringa).

Cercate di prevedere cosa stampera' il COMMODORE 64 dopo il RUN del programma che segue:

```
10 READ A$,B$,C$
20 DATA 50,60,70
30 PRINT A$+B$+C$
```

Se vi aspettavate 180 sara' meglio riprendere il primo paragrafo di questo capitolo e..... rimeditare almeno per due giri.
Se fosse necessario rileggere tutti i dati piu' di una volta si ricorrera' al comando

RESTORE (da non confondere col tasto [RESTORE])

Con questa istruzione il COMMODORE 64 azzera l'indice che gli permetteva di conoscere il prossimo dato da leggere (puntatore) e lo riporta all'inizio in modo che la successiva richiesta READ prelevera' il primo dato dall' elenco dei DATA.

```
10 READA,B,C,D
15 PRINTA,B,C,D
20 RESTORE
30 READA1,B1,C1,D1,E1,F1
35 PRINTA1,B1,C1,D1,E1,F1
40 DATA1,2,3,4,5,6
```

Si e' cosi' risparmiata una istruzione DATA 1,2,3,4
Ricordatevi di non inserire istruzioni sulla stessa riga dei dati perche' sarebbero giustamente interpretati come dati errati e si avrebbe il solito ?SYNTAX ERROR IN..

ISTRUZIONI DI SALTO

Sono utili per provocare l' esecuzione di una parte di programma che si trova lontana dal punto in cui sta avvenendo l' elaborazione, questa prosegue a quella parte, poi puó ulteriormente essere spostata a piacimento. Il comando e':

GOTO "NUMERO DI LINEA."

```
10 PRINT"DA LINEA █10■ VADO A LINEA 60":GOTO60
20 PRINT"█DA LINEA █20■ VADO A LINEA 70":GOTO70
30 PRINT"█DA LINEA █30■ VADO A LINEA 80":GOTO80
60 PRINT"█SONO IN LINEA █60■ VADO A LINEA 20":GOTO20
70 PRINT"█SONO IN LINEA █70■ VADO A LINEA 30":GOTO30
80 PRINT"█SONO IN LINEA █80■ E MI ARRESTO"
```

L'esempio e' molto semplice e provoca continui salti (anche se... inutili).
Un esempio piu' pratico si puo' avere con l'utilizzo dell'istruzione:

ON <espr.- var>. GOTO "NUM. LINEA","NUM. LINEA", ...

In questo caso il salto e' condizionato dall'espressione o variabile posta tra ON e GOTO, infatti se questa vale 1 l'elaborazione continua al primo num. di linea, se vale 2 al secondo num. di linea, se 3 al terzo, etc.

```
4 A$="QUESTO E' IL COLORE PRESCELTO"
5 PRINT"] COLORE NERO    █1■"
6 PRINT" COLORE BIANCO █2■"
7 PRINT" COLORE ROSSO  █3■"
8 PRINT" COLORE VERDE  █4■"
9 PRINT" COLORE GIALLO █5■"
10 PRINT" FINE          █6■"
11 INPUT"BATTERE UN NUMERO INTERO TRA 1 E 5";A
15 ON A GOTO 20,30,40,50,60,70
20 PRINTCHR$(144),A$:GOTO5:REM CHR$(   )E' USATA
21 REM  PER CAMBIARE IL COLORE AL CARATTERE
30 PRINTCHR$(  5),A$:GOTO5
40 PRINTCHR$( 28),A$:GOTO5
50 PRINTCHR$( 30),A$:GOTO5
60 PRINTCHR$(158),A$:GOTO5
70 STOP
```

Questo esempio da' un'idea di come si possa comporre un quadro di comando (correntemente detto MENU') per passare alla elaborazione di diverse parti di un programma.

- Se A fosse negativo il COMMODORE 64 provoca ?ILLEGAL QUANTITY ERROR e si arresta l'esecuzione del programma.
- Se A = 0 oppure ad un valore superiore ai numeri di linea previsti nell'istruzione ON ..... GOTO l'elaborazione prosegue alla linea successiva (provate a battere 0 oppure 10 alla richiesta del programma, constaterete una nuova stampa della riga 20 senza cambio di colore).
- Se A non e' intero, viene presa in considerazione la sola parte intera.

```
10 INPUT"ASSEGNARE UN NUMERO ";A
20 INPUT"QUADRATO 1 CUBO 2 INVERSO 3 FINE 4";B
30 ON B GOTO 50,60,70,80
40 GOTO30
50 PRINT"IL QUADRATO DI ";A;"E'";A↑2:GOTO10
60 PRINT"IL CUBO DI ";A;"E'";A↑3:GOTO10
70 PRINT"L'INVERSO DI ";A;"E'";1/A:GOTO10
80 STOP
```

In questo caso qualsiasi numero positivo superiore a 3.999..... riporta l'elaborazione al punto di partenza.
Cerchiamo di operare in modo che la scelta sia legata allo stesso numero assegnato: eliminate la riga 20 e modificate la 30:

```
30 ON A/100 GOTO 50,60,70,80
```

in tal modo per A inferiore a 100 e superiore a 400 si avra' richiesta di un altro dato;
per A tra 100 e 199 si otterra' il quadrato;
per A tra 200 e 299 si otterra' il cubo;
per A tra 300 e 399 si otterra' l'inverso;
per A tra 400 e 499 si arrestera' il programma.
Al posto di A/100 si potrebbe sostituire qualsiasi altra espressione calcolata in modo opportuno, il COMMODORE 64 la calcolera' e ne considerera' la parte intera.
L'uso di GOTO e' pero' veramente utile accoppiato alla possibilita' di eseguire delle scelte logiche:

IF <espr.> THEN "altre istruzioni"

- <espr.> puo' essere una espressione logica
- "altre istruzioni" e' l'insieme delle istruzioni che DEVONO essere eseguite SOLAMENTE se l'"espr." risulta VERA; viceversa l'elaborazione continua alla riga successiva.
- se <espr.> non contiene confronti logici essa viene eseguita con lo zero:
 in quest' ultimo caso le istruzioni scritte dopo lo THEN vengono eseguite solamente se l' espressione risulta diversa da zero, viceversa l' elaborazione continua alla riga successiva.

ESPRESSIONI CORRETTE	ESPRESSIONI ERRATE
IF A > 0 THEN GOTO 200	IF 5=A THEN X=Y
IF A >= 0 THEN 200	IF A <=> B THEN 100 (inutile)
IF A-B/C < 0 GOTO 200	
IF A$ = "Q" THEN PRINT A$:GOTO 100	
IF A THEN 1000	

proviamo ad usarlo per evitare errori che provocherebbero l'interruzione del programma

```
10 INPUT "BATTI 2 NUMERI";A,B
20 IF B=0 THEN GOTO 50
30 PRINT"IL QUOZIENTE TRA ";A;"E";B;"E'";A/B
40 STOP
50 PRINT"OCCHIO CHE B DEVE ESSERE ><0":GOTO 10
```

Queste 5 righe di programma potrebbero controllare che lo stesso venga eseguito senza arrestarsi per errore nel caso di divisore molto piccolo od uguale a 0; infatti in questo caso il quoziente provocherebbe un OVERFLOW.
Come avrete potuto constatare il salto da riga 20 a riga 50 (e quindi l'esecuzione di quest'ultima) avviene solamente se e' stato assegnato B = 0; viceversa il quoziente viene eseguito ed il programma si arresta. Se la riga 40 fosse stata tralasciata si avrebbe, successivamente all' esecuzione della riga 30 anche l'esecuzione anche della riga 50, poi il ritorno alla riga 10 (causato dal GOTO 10) e cio' sarebbe errato dal punto di vista della logica.
Anche l'esecuzione della radice quadrata di un numero

negativo e' impossibile e quindi e' bene premunirsi contro eventuali risultati negativi per i quali si vuol estrarre la radice quadrata:

```
10 INPUT"RADICE DA ESTRARRE";R
20 IF R < 0 THEN R=-R:GOTO 50
30 PRINT"IL CALCOLO DA'";SQR(R)
40 STOP
50 PRINT"SPIACENTE MA HO CALCOLATO"
60 PRINT"LA RADICE DEL ▒VALORE ASSOLUTO":GOTO 30
```

L'uso dell'istruzione IF e' utile anche per il confronto delle stringhe.
Ricordiamo quanto gia' detto in precedenza sulle uguaglianze, disuguaglianze tra parole:

- due parole sono uguali se:
  formate dallo stesso numero di lettere corrisponden-
  temente uguali ed ugualmente disposte;
- due parole sono diverse se:
  composte da lettere diverse,
  composte da lettere uguali diversamente disposte
  una delle due e' solo la prima parte della seconda.

Vediamo un esempio:

```
5 PRINT"▯"
10 PRINT"QUESTO PROGRAMMA FUNZIONA SOLO"
20 PRINT"SE INTRODUCI ESATTAMENTE LA PAROLA D'ORDINE"
30 INPUT"▒ PAROLA D'ORDINE■";A$
40 IF A$="QUELLO CHE VUOI" THEN 60
50 PRINT"▒SPIACENTE NON E' LA PAROLA D' ORDINE■":GOTO10
60 REM QUI COMINCIA IL PROGRAMMA VERO E PROPRIO
```

Questo esempio e' facilmente superabile con un LIST che permettera' di leggere la parola d'ordine (QUELLO CHE VUOI), ma rende l'idea di come il COMMODORE 64 puo' essere usato anche per programmi discorsivi.

```
10 INPUT "DIMMI IL TUO NOME ";A$
30 IF A$="LUISA" THEN 110
40 IF A$="FRANCESCO" THEN 110
50 IF A$="MARCO" THEN 110
70 IF A$="MARICA" THEN 110
80 PRINT "IL TUO NOME NON E' COMPRESO"
85 PRINT "TRA GLI OPERATORI USUALI"
90 PRINT "SEI SICURO DI CONOSCERE IL PROGRAMMA?"
```

```
95 INPUT "▮SI▮ ▮NO▮";B$
100 IF B$="SI" THEN PRINT "OK ";A$" BENVENUTO TRA I COMMODORIANI": GOTO 200
108 GOTO 3000 : REM ISTRUZIONI
110 PRINT "CIAO ";A$" SONO CONTENTO DI COLLABORARE CON TE"
115 PRINT "COMINCIAMO SUBITO ▮SI▮ O VUOI LE ISTRUZIONI ▮NO▮?": GOTO 95
200 REM DA QUI COMINCIA IL PROGRAMMA
```

Cercate di memorizzare il programma ora visto e battete anche il successivo:

```
10 INPUT "BATTI QUATTRO NOMI ";A$,B$,C$,D$
20 PRINT "ORA CERCHERO' DI METTERLI"
30 PRINT "IN ORDINE ALFABETICO"
40 I=0 : REM QUESTO INDICE RESTA 0 SE"
50 REM I NOMI SONO GIA' IN ORDINE ALFABETICO"
55 REM INIZIA UN CICLO DI CONFRONTI TRA A$,B$,C$,D$
60 IF A$>B$ THEN F$=A$: A$=B$: B$=F$: I=1
70 IF B$>C$ THEN F$=B$: B$=C$: C$=F$: I=1
80 IF C$>D$ THEN F$=C$: C$=D$: D$=F$: I=1
90 REM FINE DEI CONFRONTI-SE I=0 TUTTO E' OK
100 REM SE I=1 E' STATO ESEGUITO ALMENO UNO SCAMBIO
110 IF I=1 THEN GOTO 40
120 REM STAMPA RISULTATI
130 PRINT "I NOMI IN ORDINE ALFABETICO SONO: "
140 PRINT A$: PRINT B$: PRINT C$: PRINT D$
150 PRINT : GOTO 10
```

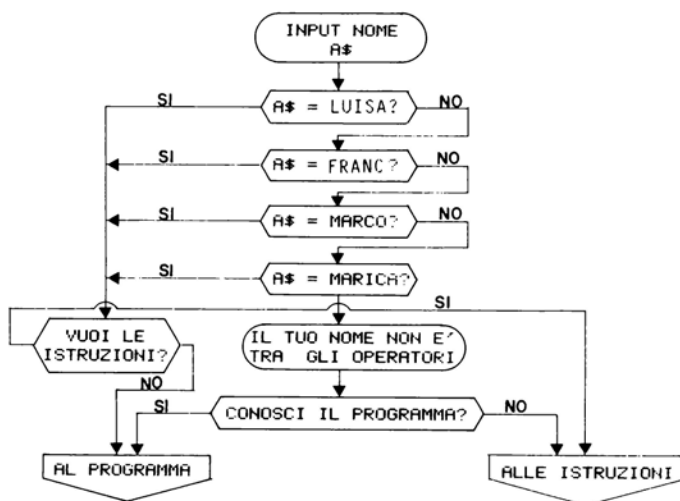



FIG.8. Schema a blocchi

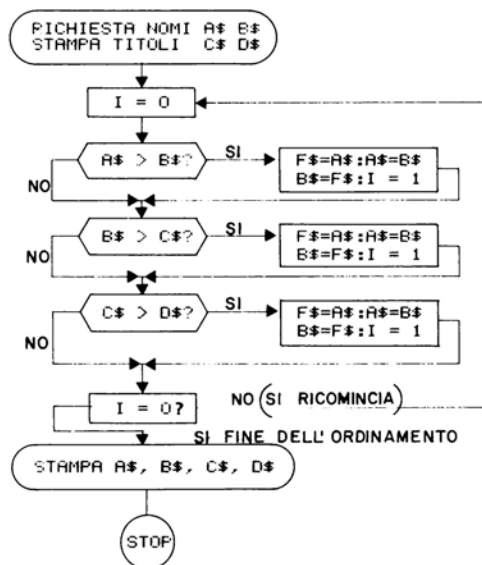



FIG.9. Schema di ordinamento tra stringhe (SORT)

Anche questo programma va salvato (salvato da SAVE significa sempre memorizzato su cassetta o disco) e meditato, si puo' anche aggiungere che sara' possibile semplificarlo drasticamente quando avrete appreso altre possibilita' di utilizzo dei vettori (si risparmieranno le righe 70-80-90 anche con un numero di nomi molto elevato.
Avete visto che l'utilizzo di GOTO, ON...GOTO, IF...THEN permettono di DIRAMARE l'esecuzione del programma, cioe' di spostare l' esecuzione in varie parti contenenti procedure diverse in funzione di scelte logiche o da tastiera. Questa potente facilitazione e' anche un'arma a doppio taglio; infatti un'errata programmazione delle scelte logiche puo' portare a risultati catastrofici. Un piccolo esempio di. errore di programmazione (detto usualmente BUG o BACO anche se realmente piu' simile ad un verme che al dignitosissimo produttore di seta...) e' gia' contenuto nel penultimo programma (l' avete trovato? no...? allora provate a cercarlo). Infatti, dopo aver dato un nome diverso da quelli previsti, rispondete SE oppure NI o qualsasi altra parola alla domanda di riga 90 : constaterete che, malgrado non abbiate risposto NO, il COMMODORE 64 andra' a riga 100 come se avesse ricevuto il NO. Infatti per prevedere

risposte diverse da SI o NO e quindi risposte errate, si dovevano scrivere le seguenti istruzioni:

```
105 IF B$="NO"THEN 1000
106 PRINT"RISPOSTA NON CHIARA":GOTO 90
```

che praticamente impedivano la continuazione del programma nel caso di risposte non previste.
Questo potrebbe mostrarvi come sia necessario, diciamo OBBLIGATORIO, far precedere la stesura di un programma dallo studio accurato della sua logica. Tale studio sara' sempre accompagnato da uno schema logico di flusso (FLOW CHART) nel quale ogni passo importante sara'

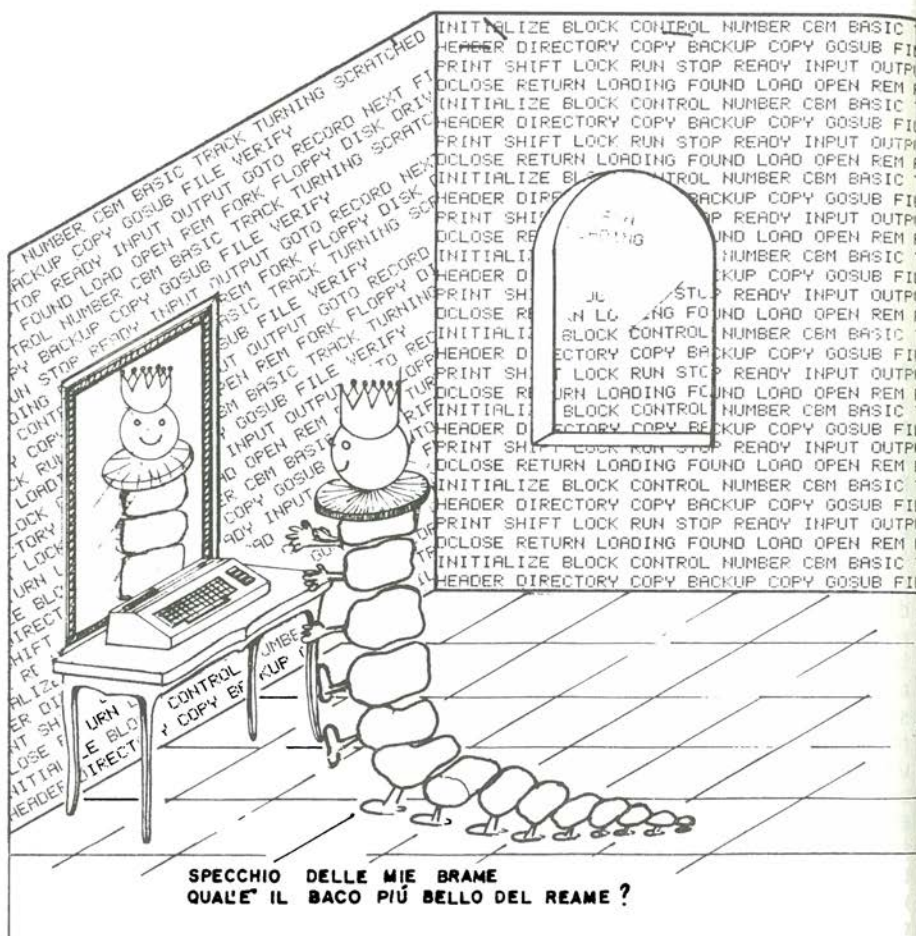



FIG.10.

racchiuso all'interno di un rettangolo, ovale, rombo o altre figure geometriche utilizzate per visualizzare le varie fasi del programma.

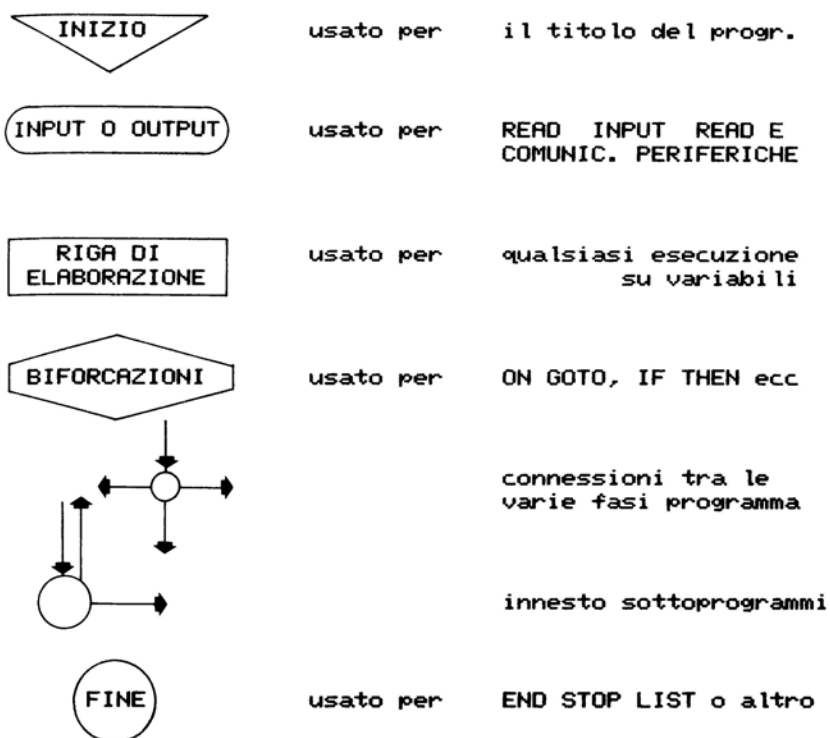



FIG.11. Simbologia usata per gli schemi a blocchi

Con tale criterio i due programmi di cui sopra sarebbero stati predisposti come
nelle figure precedenti.

PER MIGLIORARE LA COMPRENSIBILITA' DEI VOSTRI ELABORATI LOGICI POTRETE POI SBIZZARRIRVI CON COLORI, PENNARELLI E ALTRE SOTTIGLIEZZE GRAFICHE.

E' veramente importante che non sottovalutiate il consiglio di utilizzare questo metodo per eseguire lo studio della logica del programma.
Non sara' uno spreco di tempo. Anzi, dover correggere un errore di logica in un programma, quanto a perdita di tempo e... pazienza, e' gia' molto dispendioso se avete tutto il listato sotto gli occhi (cioe' se lo avete ottenuto con una stampante e potete consultarlo integralmente), diviene un pericolo per la vostra salute

(ormai gia' precaria per la lettura di questo libro) se lo dovete eseguire a brani di 23 righe sul vostro monitor!...
Tra l'altro il diagramma di flusso riesce ad evitare piu' facilmente la presenza di salti errati o la necessita' di salti che evitino lo sconfinamento con l'elaborazione in parti del programma non inerenti al procedimento di elaborazione.

L'istruzione IF...THEN puo' essere ulteriormente complicata con la verifica all' interno della stessa istruzione di piu' di una condizione , queste possono essere verificate:

- Contemporaneamente IF <...> AND <...> THEN
- Almeno una delle due IF <...> OR <...> THEN

L' esecuzione di due verifiche contemporanee (AND in questo caso significa "E CONTEMPORANEAMENTE") e' un momento di programmazione pericoloso:

IF A > 0 AND A < -5 THEN GOTO 900

non ha senso in quanto A non potra' mai essere contemporaneamente positivo e minore di -5, percio' il GOTO 900 non verra' mai eseguito (il bacherozzolo in questo momento sarebbe gia' impegnato in una sonora risata)
Sarebbe corretto

IF A > 0 AND A < 5 THEN GOTO 900

nel caso di A compreso tra 0 e 5.

Le verifiche in alternativa (OR significa oppure) richiedono che almeno una delle condizioni sia soddisfatta per l'esecuzione delle istruzioni contenute nella stessa riga.

IF A > 0 OR A < -5 THEN GOTO 900

verra' eseguito il GOTO quando A non assume valori compresi tra -5 e 0, infine:

IF A > 0 OR A < 5 THEN GOTO 900

non ha senso in quanto A < 5 contiene anche A < 0 e quindi CONTRADDICE la prima espressione che richiederebbe A > 0 (ovviamente anche la prima contraddice la seconda) percio' qualunque sia il valore

di A si otterra' sempre l'esecuzione del GOTO 900.
Poiche' questi ultimi concetti sono complicati provate i programmini seguenti in tutte le loro varianti:

```
10 INPUT "VALORE DI A";A:IF A=9 THEN STOP
20  IF  A>0  AND  A<5  THEN  PRINT  "E'  PASSATO  DAL
THEN":GOTO10
30 PRINT" NON E' PASSATO"
```

Il passaggio avviene per a tra 0 e 5 (estremi esclusi)
Cambiate la riga 20:

```
20  IF  A>0  AND  A<-5  THEN  PRINT  "E'  PASSATO  DALLO
THEN":GOTO10
```

E non c'e' valore di A che riesca a passare
Cambiate la riga 20:

```
20  IF  A>0  OR  A<-5  THEN  PRINT  "E'  PASSATO  DALLO
THEN":GOTO10
```

Passano tutti i valori di A esterni all' intervallo da -5 a 0 co;presi gli estremi.
Cambiate la riga 20 per l'ultima volta in:

```
20  IF  A>0  OR  A<5  THEN  PRINT  "E'  PASSATO  DALLO
THEN":GOTO10
```

Fate infine attenzione a non confondere AND OR usati come preposizioni nella istruzione IF...THEN dalle istruzioni Basic AND OR NOT usati quali operatori nell' algebra Booleana, infatti in tal caso questi operatori vengono applicati a variabili reali od intere e danno un risultato secondo le norme esposte piu' avanti.

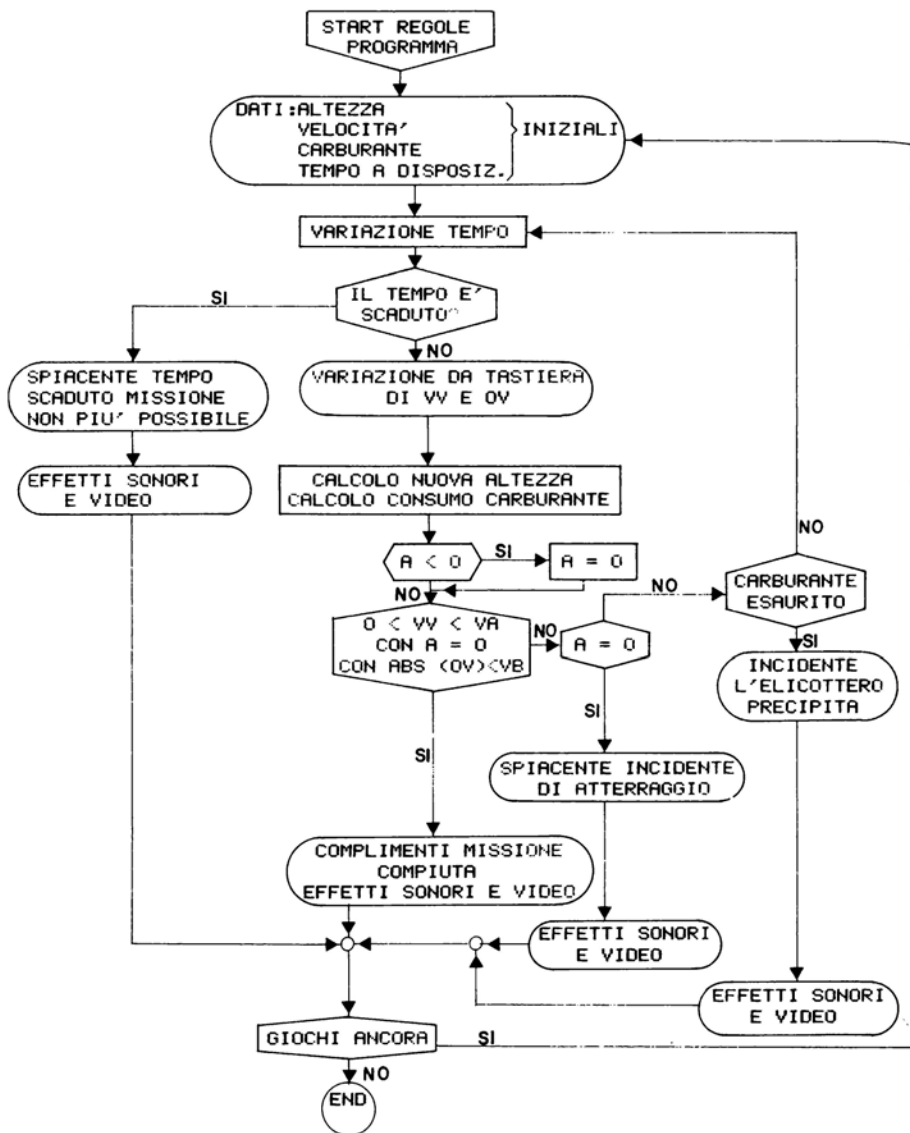



FIG.12. Schema a blocchi programma elicottero

Supponete di voler controllare l'atterraggio di un elicottero, perche' lo stesso avvenga correttamente si devono verificare tre condizioni:

- la velocita' di discesa verticale deve essere piccola, ad esempio inferiore a VA e naturalmente non dovra' essere negativa (altrimenti il velivolo sale).
- l'altezza A dell'elicottero dal suolo deve diminuire fino ad essere nulla.
- la velocita' orizzontale OV deve essere anch'essa piccola (inferiore a VB) sia positiva che negativa.

Il tutto e' esprimibile con le seguenti disuguaglianze:

$$0 < VV < VA$$
$$A = 0$$

$OV < VB$ se positiva
$OV > -VB$ se negativa

l' ultima espressione e' equivalente ad:

$$ABS\ (OV) < VB$$

dove ABS (OV) e' una funzione (valore assoluto) che toglie il segno alla variabile OV e quindi consente che il confronto (con VB) avvenga esclusivamente tra numeri positivi.
Supponiamo che attraverso alcuni tasti della tastiera voi possiate variare i tre valori VV,A,OV durante il volo, vediamo come puo' essere controllato l'atterraggio:
- se atterrate (A=0) con OV o VV superiori ai valori consentiti avverra' un incidente di atterraggio,
- ogni ciclo consumera' carburante percio' se lo stesso si esaurisce si avra' un incidente (l'elicottero precipita);
-se infine il tempo si esaurisce la missione non e' piu' possibile.
Il tutto puo' essere schematizzato come nella figura n...(FLOW CHART).........

Non e' ancora possibile comporre un intero programma perche' alcune funzioni Basic non sono state introdotte; pertanto si e' realizzata una sequenza dei controlli e si sono indicate con tante REM seguite dal loro titolo le sezioni che qui non vengono sviluppate.

```
99 REM
100 REM                    SEZIONE DI PRESENTAZIONE GIOCO  E REGOLE
101 REM
200 REM                       ASSEGNAZIONE DATI(CON READ &CLOSEDATA)
201 REM
299 REM                                               INIZIO DEL GIOCO
300 TP=TP-10 :REM                           IL TEMPO IN SECONDI VIENE
301 REM                                    DECREMENTATO DI 10 SECONDI
310 IF TP=0 THEN 700 :REM       CONTROLLA SE IL TEMPO E' SCADUTO
399 REM
400 REM                              SEZIONE DI VARIAZIONE WV ED OV
401 REM
500 REM          CALCOLO NUOVA ALTEZZA E AGGIORNAMENTO CARBURANTE
501 REM
597 REM
598 REM                                   SEZIONE DI CONTROLLO VOLO
600 IF A< 0 THEN A=0:REM  PERCHE' L'ALTEZZA PUO' ESSERE NEGATIVA
604 REM
605 REM       SE LA RIGA 610 E' SODDISFATTA IN TUTTE LE SUE PARTI
606 REM                     L' ATTERRAGGIO E' PERFETTAMENTE RIUSCITO
610 IF WV > 0 AND VV < VA AND A=0 AND ABS(OV) < VB THEN 800
614 REM
620 IF A = 0 THEN PRINT "SPIACENTE ATTERRAGGIO NON RIUSCITO!"
621 GOTO 1000
624 REM
625 REM    LA RIGA 630 VIENE ORA ESEGUITA SOLO SE A DIVERSO DA 0
626 REM                                    CIOE' ELICOTTERO IN ARIA
630 IF CR=0 THEN PRINT"SPIACENTE CARBURANTE ESAURITO":GOTO1100
635 REM C'E'CARBURANTE :IL GIOCO CONTINUA CON ELICOTTERO IN VOLO
640 GOTO 300
700 PRINT "SPIACENTE IL TEMPO E' ESAURITO"
701 PRINT "■LA MISSIONE ORMAI NON E' PIU' POSSIBILE":GOTO1200
800 PRINT "■COMPLIMENTI: L'ATTERRAGGIO E' PERFETTAMENTE RIUSCITO"
900 REM                              EFFETTI PER BUON ATTERRAGGIO
901 GOTO 1200
1000 REM                          EFFETTI PER CATTIVO ATTERRAGGIO
1001 GOTO 1200
1100 REM                    EFFETTI DELL' ELICOTTERO CHE PRECIPITA
1101 GOTO 1200
1198 REM
1199 REM                     SCELTA PER FINE O  RIPRESA DEL GIOCO
1200 Q$=""
1210 PRINT "■1■ PER GIOCARE ANCORA"
1220 PRINT "■2■ PER FINIRE"
1230 GET Q$: IF Q$="" THEN 1230: REM      ATTESA DI UN CARATTERE
1240 IF Q$="1" THEN 200
1250 IF Q$="2" THEN PRINT "CIAO ALLA PROSSIMA VOLTA": STOP
1260 GOTO1200:REM RITORNO PER QUALSIASI CARATTERE DIVERSO DA 1,2
```

L'istruzione IF consente anche di contare (cioe' di far contare il Computer) ed eseguire la stessa istruzione o

lo stesso gruppo di istruzioni un certo numero di volte, passando poi,appena raggiunto il n. desiderato altre istruzioni.

```
10 I=0: PRINT "▯"
20 I=I+1: PRINT "QUESTO E ' IL CICLO N ";I
30 IF I < 21 THEN 20
40 PRINT "HO FINITO. ADESSO I VALE ";I
50 END
```

oppure cose piu' simpatiche:

```
5 T=5 : X=10: L=20
10 PRINT "APPENA VEDI IL CARATTERE"
20 PRINT "PREMI LA BARRA SPAZIATRICE"
30 Y=INT(RND(0)*200)
40 I=0
50 I=I+1: IF I < Y THEN 50
60 Q=0: Q$=""
70 GET Q$: PRINT "▯ ▯";: Q=Q+1: IF Q$="" THEN 70
80 IF Q < T THEN PRINT "BRAVISSIMO SEI UN FULMINE"
90 IF Q < X AND Q >= T THEN PRINT "RIFLESSI PRONTI"
100 IF Q < L AND Q >= X THEN PRINT "CONCENTRATI PUOI MIGLIORARE"
110 IF Q >= L THEN PRINT "ATTENZIONE PER GIOCARE DEVI SVEGLIARTI"
115 Z$=""
120 PRINT "PREMI RETURN PER RIPROVARE":
130 INPUT Z$
140 PRINT "▯": GOTO 10
```

In questo caso si e' usata la variabile Y per generare un ciclo di ritardo variabile ogni volta proprio in funzione del valore di Y e la funzione INT (RDN (0)*900) per generare un numero intero tra 0 e 900 (infatti RDN (0) genera un numero tra 0 ed 1 che poi viene moltiplicato per 900 e depurato dalla parte decimale;

nella riga 40 l'indice I viene azzerato,
nella riga 50 I diviene 1 (la prima volta) poi il test se I < Y consente di ritornare alla riga 50 dove I diviene 2 e poi 3...fino a quando I uguaglia il valore di Y e si passa a riga 60 e successive.
In riga 70 si avra' un altro ciclo: l' istruzione GET Q$ preleva un carattere dal buffer di tastiera, stampa un carattere reverse sullo schermo, incrementa la variabile Q e controlla il carattere prelevato: se questo non c'e', ossia vale "", esegue un salto alla riga 70 cioe' riesegue in ordine le istruzioni indefinitamente; l'

uscita dal "loop" si avra' solamente dopo aver premuto un qualsiasi carattere da tastiera.
Esiste nella programmazione col Basic un metodo piu' semplice per realizzare un ciclo come in riga 40-50 ed e' l' istruzione di LOOP.

ISTRUZIONI DI LOOP

FOR <V> = <n1> TO <n2> STEP <n3>

per semplicita' vediamo prima un esempio:

```
10 FOR V = 3 TO 26 STEP 2: PRINT V
70 NEXT V
```

dove V e' una variabile che assume tutti i valori interi dispari da 3 compreso a 25 compreso cioe' 3,5,7,9,... in quanto comincia da 3 e con passo (STEP)= 2 prosegue ad incrementare fino a che il suo valore supera 26.
Riferendoci all'esempio precedente sarebbe l'equivalente di:

```
10 V=3
20 REM ISTRUZIONI VARIE DI ELABORAZIONE COME PRINT V
30 V=V+2
70 IF V < 26 THEN 20
```

Tutto l'insieme di istruzioni eseguite fino a che non si trova l'istruzione NEXT V viene ripetuta ciclicamente incrementando il valore di V (praticamente NEXT esegue il confronto tra V e 26). L'elaborazione, finito il ciclo proseguira' alla istruzione successiva al NEXT V ed il valore di V sara' quello dato dall'ultimo incremento che ha provocato la fuoriuscita dal ciclo (cioe' nel nostro caso V=27).
Dopo NEXT la variabile puo' anche non essere indicata.

- Se non specificate STEP il passo scelto sara' 1,
- il passo potra' anche essere un numero non intero;
- potrete usare estremi anche negativi o costituiti da espressioni numeriche;
- potrete contare a rovescio se userete un valore negativo dopo STEP;
- se gli estremi coincidono il ciclo avviene sempre anche se una sola volta.

GIUSTI	ERRATI
FOR I%=20+A TO B	FOR I=20 TO 40 STEP -1
FOR V =-6 TO-20 STEP -2	FOR A%=20 STEP A+B
FOR T =-6 TO 20 STEP 3.14	NEXTC:FOR C=5 TO 6
FOR Z = A-5 TO A+5	FOR I=1:NEXT I
FOR Z = ABS(A)TO B STEP D	FOR H=5 TO 2 !!!!!!
FOR H=LOG(A) TO LOG(B) STEP LOG(C)	

E' importante tenere presente che il ciclo FOR...NEXT deve sempre essere completato attraverso l'istruzione NEXT.
Questa puo' anche essere multipla:

```
10 FOR I = 1 TO 20
20 IF I <= 10 THEN PRINT "I MINORE DI 10":NEXT
30 PRINT "I MAGGIORE DI 10":NEXT
40 STOP
```

In questo caso dalla fine della riga 20 si ritorna alla riga 10 fintanto che I e' < 10 poi il ritorno avverra' attraverso il NEXT in riga 30.
In caso di istruzioni IF ricordatevi di verificare l'esistenza del NEXT anche successivamente a quella riga in istruzioni non condizionate, perche' non e' detto che il next della riga IF venga sempre eseguito.
Togliete ora la riga 30 e date RUN; vi accorgerete che l'elaborazione si arresta per I=11, infatti se in un ciclo manca il NEXT lo stesso viene eseguito solo una volta volta (nell' esempio I arriva a 10 perche' trova il NEXT sulla riga 20, poi al ciclo successivo passa a riga 30 e non trovando il next si arresta).
Quando il Basic incontra un FOR I memorizza la sua posizione ed incontrando NEXT ripetera' l'esecuzione del programma all'inizio del ciclo.
I pericoli di confusione nascono soprattutto quando all'interno del ciclo esistono dei salti all'esterno e degli altri cicli subordinati al primo.
- I salti si possono eseguire a patto che dall'esterno si ritorni poi ordinatamente nel ciclo ogni volta;
- non eseguite mai salti che interrompano il ciclo, in questo caso il primo next non seguito da specifica di variabile verra' usato dal calcolatore come chiusura del ciclo precedente e cio' provochera' sicuramente un errore logico, nel caso piu' fortunato invece, ossia incontro con un NEXT dotato di variabile si avra' segnalazione di errore,se infine il NEXT avra'la stessa variabile ma sara' quello di un altro ciclo dovrete ingaggiare una caccia allo errore di non facile vittoria.

```
10 FOR I = 1 TO 5
20 IF I = 1 GOTO50
30 PRINT I;"SUPERIORE A 1"
40 NEXT I: STOP
50 PRINT"QUESTO E'·IL PRIMO GIRO" :NEXT
```

Prudenzialmente e' preferibile rinviare sempre allo stesso NEXT cioe':

```
50 PRINT"QUESTO E' IL PRIMO GIRO":GOTO40
```

Errato invece e':

```
50 PRINT"QUESTO E' IL PRIMO GIRO":GOTO10
```

infatti in questo caso ad ogni giro si apre un ciclo senza mai chiuderlo (e I resta sempre 1). Se provate a far girare il programma vi renderete conto di essere incappati in un errore che in gergo viene indicato con LOOP, cioe' avete programmato il Computer in modo da non consentirgli piu' di arrestare l' elaborazione. Di questi errori se ne possono commettere a volonta' sia con l' istruzione FOR NEXT che con l' istruzione IF THEN:

```
10 I=0
20 I=I+1
30 IF I>0THEN 20 (si arresta per overflow ma dopo anni!)
```

oppure:

```
10 FOR I= 20 TO 30 STEP-1:PRINT I:NEXT I
```

Nel caso di apertura di piu' cicli successivi ricordare che IL PRIMO APERTO deve ESSERE L'ULTIMO CHIUSO, IL SECONDO ... IL PENULTIMO e cosi' via.
Evitate sempre per logica o per disattenzione, di programmare cicli composti in modo da intersecarsi;

```
10 FOR I = 1 TO 3: PRINT I: PRINT "CICLO ESTERNO"
20 FOR K = 0 TO 9
30 PRINT K;: NEXT K
40 PRINT: PRINT: NEXT I
```

Per chiarezza nella stesura del programma potete anche ricorrere ad artifici spostando l'inizio delle righe all'interno o distinguendo i cicli con colori diversi

```
10 FOR I = 1 TO 3
20 FOR K = 1 TO 3
30 FOR H = 1 TO 3
40 PRINT I;K;H;
50 NEXTH
60 PRINT: NEXT K
70 PRINT "******": NEXT I
```

Il orecedente programma verra' schematizzato nel diagramma di flusso come segue:

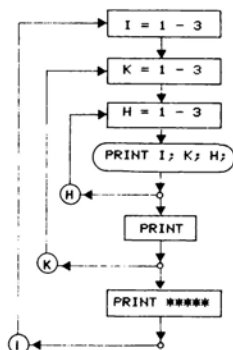



FIG.13. Schema a blocchi di cicli for next nidificati

Utilizziamo le istruzioni viste fino ad ora per giocare col COMMODORE 64:

```
3 N%=0:V%=0
5 N%=RND(0)*7
10 INPUT"INSERIRE UN NUMERO INTERO FRA 1 E 6";Q%
15 N=N+1
20 IF N%=Q% THEN GOTO 200
100 PRINT" SPIACENTE IL TUO NUMERO"
110 PRINT" NON COINCIDE CON IL MIO"
120 GOTO 300
200 PRINT"HAI AVUTO UNA FORTUNA SFACCIATA"
210 PRINT "AVEVO PENSATO ";N%;" ANCH'IO"
220 V=V+1
300 PRINT "ORA LA SITUAZIONE E' LA SEGUENTE"
310 PRINT "NUMERO DI TENTATIVI ";N
320 PRINT "NUMERO DI VITTORIE  ";V
330 PRINT "PERCENTUALE         ";V/N*100
340 IF V < N/6 THEN PRINT"SEI SFORTUNATO":GOTO 400
350 IF V < N/3 THEN PRINT"MICA MALE PER UN UMANO":GOTO 400
360 IF V > N/2 THEN PRINT"HO IL SOSPETTO CHE TU USI UN'ALTRO COMPUTER"
400 PRINT"C PER CONTINUARE"
410 PRINT"F PER FINIRE"
420 INPUT Q$: IF Q$="C" THEN 5
430 IF Q$="F" THEN PRINT "CIAO ":STOP
440 GOTO 400
```

SOTTOPROGRAMMI

Spesso risulta necessario eseguire una o piu' istruzioni tra loro uguali o simili, ripetendole continuamente nel programma ed occupando cosi' memoria programma inutilmente.
Un metodo efficacissimo per evitare questo inconveniente e' il ricorso a sottoprogrammi.
Un sottoprogramma (chiamato subroutine) non e' altro che una sequenza di istruzioni che terminano con l' istruzione Basic RETURN, ossia l' ordine di ritorno al programma principale.
Nel programma principale il salto al sottoprogramma (detto anche la chiamata del sottoprogramma) avviene con una istruzione:

GOSUB num. di linea

oppure

ON < VAR > GOSUB num. di linea, num. di linea...

Come avete visto per l' istruzione ON GOTO anche l' ultima delle due elencate decide il salto al sottoprogramma avente inizio al numero di linea corrispondente al valore intero della variabile, cioe' salta al primo n. di linea ve <VAR>=1, al secondo se <VAR>=2 ,mentre se <VAR> e' uguale ad un numero ed il corrispondente n. di linea non c'e' o se vale zero l' elaborazione prosegue alla linea programma successiva.
Il sottoprogramma e' formato da un insieme di linee che e terminera' sempre con l'istruzione RETURN che non va confuso col tasto [RET], questa istruzione va proprio battuta da tastiera lettera dopo lettera (o anche nella forma abbreviata re [SHIFT] t.
Esiste una sostanziale differenza tra l'istruzione GOSUB e l'istruzione GOTO, infatti quando viene eseguito il GOSUB il Basic del COMMODORE 64. memorizza la posizione dell'istruzione nel numero di linea ed il numero di linea stesso, in modo che, trovando un RETURN, l'elaborazione vi ritorna nuovamente. La mancanza dell'istruzione RETURN alla fine del sottoprogramma provocherebbe una grossa confusione nella memoria durante l'esercizio del programma, anche col rischio di segnalazione di errore. Provate ad esempio a provocare il rientro nel programma principale con un GOTO al posto del RETURN (fatelo questa volta e poi non fatelo piu' !).

```
10 PRINT "LANCIO DEL SOTTOPROGRAMMA"
20 GOSUB 300
50 GOTO 10
300 PRINT "ESECUZIONE DEL SOTTOPROGRAMMA":GOTO 50
```

Dopo vari LANCIO ESECUZIONE troverete un ? OUT OF MEMORY ERROR IN 300.
Vediamo cosa e' avvenuto:
- in riga 20 il Basic del COMMODORE 64 trovando l' istruzione GOSUB 300 memorizza il numero di linea e la posizione dell' istruzione in un registro detto STACK e poi provoca il salto, solamente quando si incontra l' istruzione return il Basic recupera dallo STACK i dati memorizzati e li utilizza per riportarsi alla posizione precedente dove riprende l' elaborazione.Se dopo un GOSUB se ne esegue un altro, i due nuovi indirizzi vengono inseriti nello STACK, spostando i vecchi dati di due posti, cio' perche' l' ultimo indirizzo memorizzato deve essere il primo in caso di prelievo, anzi quando un dato viene prelevato gli altri si spostano tutti di un posto ed il secondo si posiziona prontoper il prelievo successivo. Per avere un esempio pratico pensate ad una pila di fogli su cui si appoggiano altri fogli: voi potete appoggiarne uno, poi un'altro,poi un'altro ancora ma l' ultimo che avete appoggiato sulla pila e' anche l'unico che potete prelevare senza scomporre la pila. Poiche' anche lo STACK ha un limite di memoria di 256 bytes, risulta impossibile superare un certo numero di GOSUB una in sequenza all'altra perche' si avra' segnalazione di errore per mancanza di memoria, come e' avvenuto nel precedente programma esemplificativo (da non imitare in futuro), che ha continuamente provocato lanci di sottoprogrammi senza mai chiuderli con l'istruzione RETURN.
Come gia' detto un sottoprogramma puo' contenere il lancio di piu' sottoprogrammi che a loro volta possono utilizzare altri sottoprogrammi:

```
10 PRINT "⊐ CALCOLERO' L'AREA DI QUESTE FIGURE"
15 PRINT:PRINT
20 PRINT "RETTANGOLO      1"
30 PRINT "QUADRATO        2"
40 PRINT "TRIANGOLO RETT.3"
50 PRINT "TRIANGOLO       4"
60 PRINT:PRINT:INPUT"           QUALE ";A
70 ON A GOSUB 200,300,400,500
80 Q$ = ""
90 GET Q$:IF Q$ = "" THEN 90
```

```
100 GOTO 10
200 INPUT "BASE , ALTEZZA";B,H
210 PRINT "L'AREA DELLA FIGURA E' ";B*H
220 PRINT:PRINT: RETURN
300 INPUT "LATO ";L  •
310 PRINT "L' AREA DELLA FIGURA E' ";L↑2
320 PRINT:PRINT: RETURN
400 INPUT "BASE , ALTEZZA ";B,H
410 PRINT "L' AREA DELLA FIGURA E' ";B*H/2
420 PRINT:PRINT: RETURN
500 INPUT "LATO 1 , LATO 2 , LATO 3 ";L1,L2,L3
505 P=(L1+L2+L3)/2:S=SQR((P-L1)*(P-L2)*(P-L3))*P
510 PRINT "L'AREA DELLA FIGURA E' ";S
520 PRINT:PRINT:RETURN
```

Noterete nel programma precedente come alcune istruzioni si ripetano inutilmente piu' volte, anche se si e' ricorso alla istruzione ON GOSUB.
Inoltre e' stato inserito un ciclo di attesa nella istruzione 90 che funziona cosi':
- Preleva dal Buffer un carattere;
- se questo e' nullo ripete l'operazione;
- se e' stato premuto un carattere prosegue;
  il ciclo e' necessario per evitare che la prima stampa di riga 10 cancelli il risultato cosi' rapidamente da non consentirvi di leggerlo.
Per migliorare la forma del programma si puo' definire all'inizio una parte della frase con una stringa (ad esempio F$="L'AREA DELLA FIGURA E'") ed ordinarne ogni volta la stampa (PRINT F$), ma il programma e' ancora sintatticamente lungo. Provate a vedere le seguenti variazioni (lasciando inalterate le prime istruzioni. Vedere anche il diagramma di flusso):

```
100 GOTO 10
200 GOSUB1000
210 GOSUB1100:PRINT B*H
220 RETURN
300 INPUT"LATO ";L
310 GOSUB1100:PRINT L↑2
320 RETURN           .
400 GOSUB1000
410 GOSUB1100:PRINT B*H/2
420 RETURN
500 INPUT"LATO 1, LATO 2, LATO 3";L1,L2,L3
505 P=(L1+L2+L3)/2:S=SQR((P-L1)*(P-L2)*(P-L3)*P)
510 GOSUB1100:PRINT S
520 RETURN
1000 INPUT"■BASE,ALTEZZA■";B,H:RETURN
1100 PRINT"■■L'AREA DELLA FIGURA E'■ ";:RETURN
```

Anche questo programma e' ancora troppo lungo (ad esempio la chiamata della SUB 1100 viene fatta 4 volte); proviamo ancora a modificarlo:

```
10 PRINT " CALCOLERO' L'AREA DI QUESTE FIGURE"
15 PRINT:PRINT
20 PRINT "RETTANGOLO      1"
30 PRINT "QUADRATO        2"
40 PRINT "TRIANGOLO RETT.3"
50 PRINT "TRIANGOLO       4"
60 PRINT:PRINT:INPUT"          QUALE ";A
70 ON A GOSUB 200,300,400,500
80 PRINT"L'AREA DELLA FIGURA E' ";S:Q$=""
90 GET Q$:IF Q$ = "" THEN 90
100 GOTO 10
200 GOSUB1000:S=B*H:RETURN
300 INPUT"LATO ";L:S=L↑2:RETURN
400 GOSUB1000:S=B*H/2:RETURN
500 INPUT"LATO 1, LATO 2, LATO 3";L1,L2,L3
505 P=(L1+L2+L3)/2:S=SQR((P-L1)*(P-L2)*(P-L3)*P):RETURN
1000 INPUT"BASE,ALTEZZA";B,H:RETURN
```

Ora se volete proprio "affinare le lame" spostate i sottoprogrammi in testa al programma (infatti il COMMODORE 64 cerca i sottoprogrammi linea per linea partendo dalla prima, percio' se sono in testa si perde meno tempo), usate un tasto per uscire dal programma ed

```
10 GOTO1010
100 INPUT"BASE,ALTEZZA";B,H:RETURN
200 GOSUB100:S=B*H:RETURN
300 INPUT"LATO ";L:S=L↑2:RETURN
400 GOSUB100:S=B*H/2:RETURN
500 INPUT"LATO 1, LATO 2, LATO 3";L1,L2,L3
505 P=(L1+L2+L3)/2:S=SQR((P-L1)*(P-L2)*(P-L3)*P):RETURN
600 PRINT"BATTERE UN TASTO":Q$=""
610 GET Q$:IF Q$ = "" THEN 610
620 RETURN
1010 PRINT " CALCOLERO' L'AREA DI QUESTE FIGURE"
1020 PRINT "RETTANGOLO       1"
1030 PRINT "QUADRATO         2"
1040 PRINT "TRIANGOLO RETT.  3"
1050 PRINT "TRIANGOLO        4"
1060 PRINT"FINE             5"
1070 GOSUB600
1080 ON VAL(Q$) GOSUB 200,300,400,500,1110
1090 PRINT "L'AREA DELLA FIGURA E' ";S:PRINT
1100 GOSUB600:GOTO1010
1110 END
```

utilizzate il meccanismo di ritardo anche per prelevare il carattere che viene inserito nel Menu', (cosi' non appena premuto il num., il programma proseguira' senza attendere il [RET].

In riga 1080 e' stata inserita l'istruzione VAL (Q$) che permette a Q$ (stringa) di trasformarsi in numero (si passa dalle parole (numeriche) al valore numerico rappresentato dalle lettere), cioe' necessario per evitare di dover inserire un GET Q e vedere il programma arrestarsi per errore se al posto di un numero inavvertitamente premete una lettera. A questo proposito risulta molto inutile anche le seguenti istruzioni aggiuntive:

```
1075 IF Q$ <"1" OR Q$ >"5" THEN PRINT "DATO ERRATO"
1076 PRINT "RICOMINCIARE":GO TO 1070
```

ULTIME RACCOMANDAZIONI

- E' errato creare sottoprogrammi che si intersecano, cioe' l'ultimo chiamato deve essere chiuso da un RETURN per ritornare al penultimo, etc...;
- potete concepire dei sottoprogrammi con diverse entrate ed anche diversi ritorni, l'importante e' che controlliate molto bene che nello svolgimento il Computer non incontri piu' di una volta il RETURN di uno stesso GOSUB (o avrete segnalazione di errore):

RETURN WITHOUT GOSUB ERROR IN num. di linea

L'utilizzo dei sottoprogrammi e' fondamentale quando si debba studiare un programma complicato:
in questi casi si struttura il programma in una parte principale (MAIN) che ha la sola funzione di richiamare le parti del programma suddivise in sotto programmi, cio' avviene a volte anche se alcune parti vengono eseguite una sola volta.
Questo metodo semplifica notevolmente la formazione del programma perche' permette di affrontare la stesura per piccole dosi.

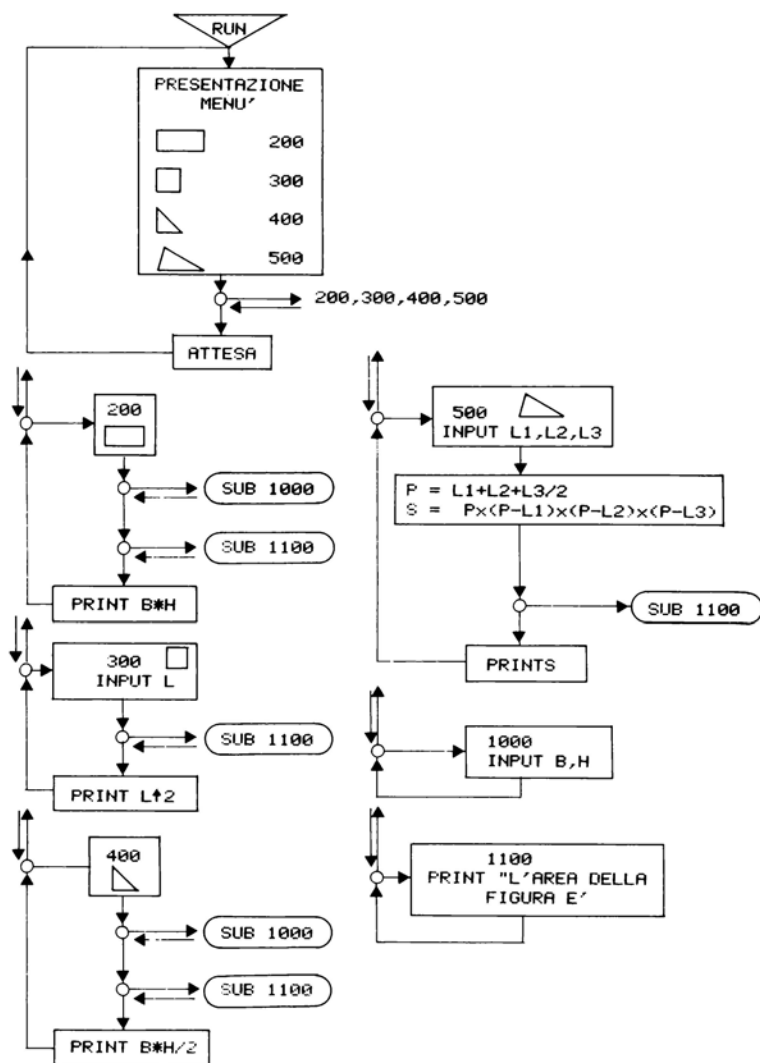



FIG.14. Schema a blocchi progr. calcolo superfici

SPAZIO PER ELUCUBRAZIONI

PEEK (V)   POKE H,K

Queste due istruzioni vengono esaminate ora in quanto permettono al programmatore di interagire direttamente con tutta la memoria del COMMODORE 64.

Non vi preoccupate.... non si rompe o guasta nulla, se il COMMODORE 64 si dovesse bloccare bastera' spegnere e riaccendere e tutto tornera' normale.
Per curiosare all'interno del COMMODORE 64 si usa la funzione PEEK (V), essa ritorna il contenuto di una delle 65536 posizioni di memoria.
Tale contenuto e' espresso in forma decimale e va da 0 a 255.
Ricordate che nel COMMODORE 64 la memoria e' raggruppata in 64 pagine di 256 celle (da 0 a 65535) di queste sono a disposizione dell'utente le seguenti:

```
1024  2023  MEMORIA VIDEO
2048  40959 MEMORIA PER BASIC O DATI PROGRAMMA
55296 56295 MEMORIA COLORE SCHERMO
53248 53295 MEMORIA PER FIGURE ANIMATE
54272 54296 MEMÓRIA PER EFFETTI SONORI
```

Se ad esempio avete appena acceso il COMMODORE 64:

```
   PRINT PEEK (53280),PEEK (53281)
```

dara' 254, 246 che corrisponde al codice colore dello sfondo e dello schermo. Con la funzione POKE H,K si puo' invece modificare il contenuto avente valore H della Kesima memoria.
Se ora volete provare a modificare i colori provate il seguente programma:

```
10 FOR I = 0 TO 15
20 FOR K = 0 TO 15
30 POKE 53280,K:NEXT:FOR H= 1 TO 50: NEXT H
40 POKE 53281,I:NEXT
```

ed avrete la possibilita' di vedere tutti i colori utilizzabili sul COMMODORE 64.
Il programma ha semplicemente cambiato di volta in volta il contenuto dei due registri colore e quindi anche il colore dello schermo con una rapidita' istantanea, tant'e' che per poter vedere la sequenza colori si aggiunge un ciclo di ritardo in riga 30 che produce solo una perdita di tempo

```
FOR H= 1 TO 50:NEXT H
```

Se alla fine risultasse difficile la lettura premete

[CTRL] 1 , oppure RUN/STOP RESTORE
contemporaneamente e il list ritorna visibile.
Se ad esempio volete simulare un effetto esplosione potrete far variare il colore video simultaneamente:

```
10 FOR I = 1 TO 14
20 POKE 53281,I:POKE 53280,I+1
30 FOR H = 1 TO 60: NEXT H
70 NEXT 1
80 POKE 53281,?:POKE 53280,?
```

dove al posto dei due ? metterete i valori del colore dello schermo iniziale (altrimenti resterebbe colorato con gli ultimi colori corrispondenti agli ultimi numeri del ciclo FOR NEXT)
Provochiamo ora la comparsa sullo schermo di un carattere:
lo schermo va da 1024 compreso a (25 righe x 40 colonne + 1023) 2023.
Cominciamo ad esempio con un carattere con comando diretto

POKE 1024,65:
POKE 55296,0 ed ecco apparire in alto il simbolo nero di picche.

```
10 FOR I = 1024 TO 2023 STEP 4
20 POKE I,65:NEXT
```

Se date RUN accadono cose strane ma non c'e' errore:
premete [SHIFT] [CLR/HOME] e poi

```
5 PRINT " < SHIFT > < CLR/HOME > "
30 FOR I= 55296 TO 56295: POKE I,0:NEXT
```

Ora col RUN avrete la sorpresa di vedere cosa e' successo.
Infatti il COMMODORE 64 prima scrive sul video i caratteri col colore di fondo,poi cambiando il colore al carattere lo visualizza.
Modificate la riga 20 con un carattere grafico diverso (es:102).

Generiamo ora una cornice:

```
10 FOR I = 1024 TO 1023 + 40
20  POKE  I,102:POKE  I+960,102:NEXT:REM  TRACCIA LA RIGA
SUP ED INF
30 FOR I = 1024 TO 1023 + 960 STEP 40
40  POKE  I,102:POKE  I+39,102:NEXT: REM TRACCIA LE RIGHE
VERT.
50 FOR I = 55296 TO 56295:POKE I,0:NEXT
```

Generiamo una scacchiera:

```
5 PRINT" < SHIFT > < CLR/HOME > "
10 FOR I= 1024 TO 2023 STEP 4
20 POKE I= 1 TO 40
30 FOR K= 0 TO 24 STEP 5
50 POKE 1023+I+40*K,102:NEXT:NEXT
100 FOR I = 55296 TO 56295:POKE I,0:NEXT
```

Se ora volete complicare il tutto ecco la scacchiera multicolore:

```
90 POKE 53280,1:POKE 53281,3
100 FOR I= 55296 to 56295:POKE I,(RND(0)*16):NEXT
```

ora provate con tutto lo schermo:

- eliminate le righe 30,40,50
- modificate 10 FOR I= 1024 TO 2023
- modificate il carattere dato dal 102 col 160.

Il programma risultera':

```
10 PRINT " < SHIFT > < CLR/HOME > ":FOR I=1024 TO 2023
20 POKE I,160:NEXT
90 POKE 53280,1:POKE 53281,3
100 FOR I=55296 TO 56295: POKE I,INT(RND(0)*16):NEXT
```

un'ultimo tocco di follia lo potrete avere inserendo le seguenti linee:

```
5 POKE 53281,3
20 POKE I,INT(160 + RND(0)*96):NEXT
```

e modificando lo 01 della POKE di linea 90 con altri valori(es:0,5,6,7,etc.)
Provate anche questo programma:

```
5 PRINT " < SHIFT > < CLR/HOME > ":GOSUB 100
6 POKE 53281,3:T=160
7 T = T+1:IF T = 255 THEN T=160
10 FOR I= 1024 TO 2023
80 F = RND(O) *16
85 F1 = RND (0) * 16
86 IF FI = F TH F1=F1/2
90 POKE 53280,F:POKE 53281,FI
100 FOR I=55296 TO 56295:POKE I,RND (0)*16:NEXT:RETURN
```

provate ora a costruire un programma che sfrutti quanto e' stato illustrato per realizzare successivi rettangoli concentrici,e con simmetria rispetto al centro.
Ricordate che per fare apparire un punto in riga R ed in colonna C bisogna eseguire una POKE K,S dove K vale 1023+C+R*40 ed S e' il numero corrispondente al simbolo grafico (vedere tabella).
Con la funzione POKE si puo' anche realizzare una figura anche se molto rudimentale rispetto a quelle ottenibili con gli SPRITE che vedremo piu' avanti)

```
10 POKE 1600,85:POKE 1639,85:POKE 1640,113
20 POKE 1641,73:POKE 1680,81:POKE 1720,81
30 POKE 1759,112:POKE 1760,113:POKE 1761,110
40 POKE 1799,113:POKE 1801,113.
50 FORI=55296 TO 56295:POKE I,2:NEXT
```

Esistono anche altre funzioni attivabili con l'istruzione POKE :come ad esempio, l'auto ripetizione dei tasti(cioe' premendo il tasto della lettera A e mantenendolo premuto la A verra' ripetuta fino a che non rilascerete il tasto).
Questo si attiva con POKE 650,128 e si disatttiva con POKE 650,64.

VETTORI E MATRICI

Poiche' il nome Vettore o Matrice ha un sapore "paurosamente matematico" per ora ragioniamo come se fossero esclusivamente tabelle.
Supponete che una ditta registri i kilometri percorsi giornalmente da uno dei suoi furgoni (diciamo il furgone A).
Potremmo indicare con A1,A2,A3,A4,A5,A6,A7 i dati di una settimana (sette variabili) ma per la settimana successiva sarebbe gia' un problema il fatto che per il COMMODORE 64 A1 ed A10, A11, A12, A13, A14 sono la stessa cosa.

prima settimana	1	2	3	4	5	6	7
	lun	mar	mer	giov	ven	sab	dom
Furgone A	100	150	170	250	260	50	30

Il COMMODORE 64 ha la possibilita' di usare una variabile AV (I) dotata di indice che si definisce con una o due lettere di cui la prima alfabetica e la seconda (non obbligatoria) alfabetica o numerica, da una

parentesi aperta, un indice (che e' una variabile numerica) ed una parentesi chiusa.
In questo modo per valori diversi dell'indice la stessa variabile puo' contenere diversi valori.
Per il calcolatore l'indice non ha limiti, pertanto nel nostro esempio potrete utilizzare A(N) anche per i giorni successivi al 7 a patto che avvertiate il COMMODORE 64 all'inizio del programma ed UNA SOLA VOLTA di quanti elementi sara' composta la vostra tabella.
L'istruzione per "dimensionare" la tabella si chiama
DIM A (N)
dove N puo' anche essere una variabile definita da tastiera o nel programma in precedenza es:

```
1 INPUT "QUANTI GIORNI CONSIDERI"
2 DIM A(G)
10 FOR R=1 TO 4
20.............
```

Se il COMMODORE 64 incontra per 2 volte il dimensionamento di una stessa variabile segnala errore:

? REDIM'D ARRAY ERROR IN....(n riga)

Se non dimensionate il vettore A, questo viene automaticamente considerato con una estensione di 11 elementi come se aveste scritto DIM A (10) (infatti da zero a 10 compresi si contano 11 numeri)

```
10 FOR R= 1TO 7
20 READ A(R):NEXT R
30 DATA 100, 150, 170, 250, 260, 50, 30.
40 FOR R=1 TO 7
50 PRINT R,A (R):NEXT R.
```

Vi e' piaciuto? Invece di inserire 7 variabili se ne usa una, e col ciclo FOR...NEXT si risparmiano anche molte istruzioni.

Se ora i furgoni fossero due si potrebbe pensare di usare una seconda colonna per il secondo furgone, allora la variabile A diviene a due indici A(R,C); il primo indica la Riga della tabella il secondo la Colonna della tabella.

prima settimana	1	2	3	4	5	6	7
	lun	mar	mer	giov	ven	sab	dom
Furgone B	111	222	333	144	166	88	0

Se nella prima colonna teniamo i percorsi del primo furgone e nella seconda colonna quelli del secondo, tutti i dati richiesti ad A(R,1) facendo variare solo R saranno i percorsi del primo furgone mentre tutti i dati richiesti ad A(R,2) facendo variare solo R saranno i percorsi del secondo furgone:

	PRIMA COLONNA FURG.N.1	SECONDA COLONNA FURG.N.2	
LUNEDI'	100	111	I^ RIGA
MARTEDI'	150	222	II^ RIGA
MERCOLEDI'	170	333	
GIOVEDI'	250	144	
VENERDI'	260	166	
SABATO	50	88	
DOMENICA	30	0	VII^ RIGA

Se invece prendo A (5,C) e vario C (la colonna)ottengo i dati 260,166 ossia le percorrenze del quinto giorno rispettivamente del primo furgone (C=1) e del secondo (C=2).
Per assegnare i dati faccio leggere prima quelli di un furgone poi quelli del secondo.

```
10 FOR R= 1 TO 7
20 READ A(R,1):NEXT R
21 FOR R=1 TO 7
22 READ A (R,2):NEXT R
30 DATA 100,150,170,250,260,50,30
41 DATA 111,222,333,144,166,88,0
```

Oppure piu' correttamente nidificando i due cicli e risparmiando cosi' istruzioni:

```
15 FOR C=1 TO 2:REM CICLO ESTERNO
10 FOR R=1 TO 7:REM CICLO INTERNO
20 READ A(R,C):NEXT R:REM CICLO ESTERNO
25 NEXT C:REM CICLO ESTERNO
30 DATA 100,150,170,250,260,50,30
31 DATA 111,222,333,144,166,88,0
```

Supponiamo ora di volere inserire per 4 settimane (S=numero di settimane) i dati di F furgoni e di stampare poi i dati per colonne (ogni colonna un

furgone), ecco come fare il programma:

```
10 INPUT"NUM.DI SETTIMANE ,NUM DI FURGONI";S,F
20 G= S*7:DIM A(G,F)
30 FOR C=1 TO F
40 FOR R=1 TO 7 * S
50 PRINT"FURGONE N^";C;"GIORNO";R
60 INPUT A(R,C)
70 NEXT R:PRINT "FINE DATI FURGONE N^";C
80 NEXT C
100 REM STAMPA
110 FOR R=1 TO S*7:PRINT R;
120 FOR C=1 TO F
130 PRINT A (R,C);
140 NEXT C:PRINT;NEXT R.
```

Potete provare con parecchi furgoni e varie settimane,l'unico incoveniente nelle tabelle sara' dato dal non perfetto incolonnamento dati, potete pero' provare ad usare la funzione TAB.ad esempio con 8 spazi(quindi max 5 furgoni per riga) dovrete allora modificare la riga 130:

```
130 PRINT TAB (8*C+) A (R,C);
```

e per rendere piu' chiara la tabella mettere in testa alla scritta un titolo del tipo sotto riportato,

```
105PRINT"<CRSV CRSV>GIORNO"TAB(7)"FURG.1"TAB(15)FURG.2";
106 PRINTTAB(23)"FURG.3"TAB34"FURG.4"
```

Questa istruzione pero' e' passibile di numerosi ragionamenti legati al n^ di furgoni, pertanto e' bene costruirla usando le funzioni di manipolazione caratteri (come vedremo piu'avanti) ad es:se i furgoni sono tre e' inutile far comparire l'ultima parte.
Ora che l'esempio ci ha dato un'idea semplice di cosa si voglia intendere con i termini matrice e vettore,vediamo di descriverle piu' accuratamente.
Una matrice e' una tabella di dati con due o piu' dimensioni.
Un vettore e' una matrice avente una sola colonna (e numero di righe qualsiasi).
Il vasto COMMODORE 64 ha poi delle possibilita' impensate:
_ potete definire matrici con piu' di due dimensioni, ad esempio la terza dimensione potrebbe essere una seconda tabella su un piano diverso dalla prima:

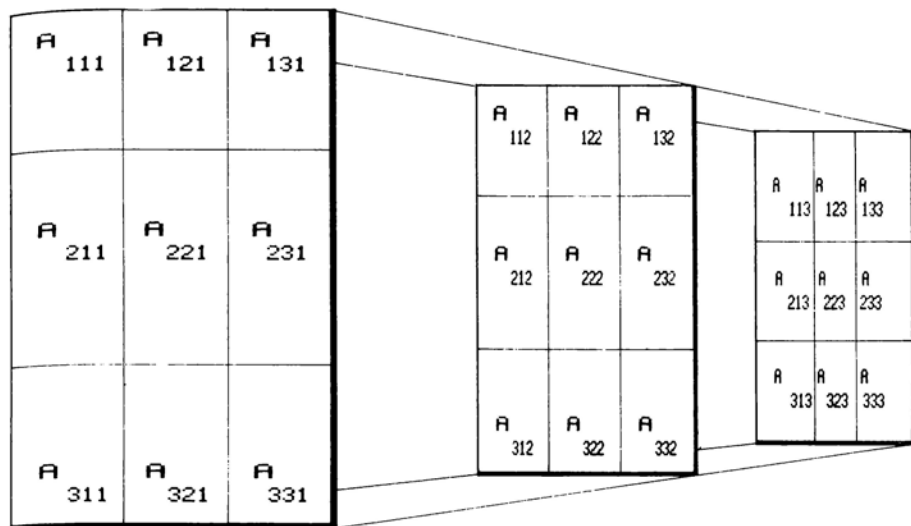



FIG.15. Esempio di matrice a tre dimensioni

E' come se la nostra ditta volesse rilevare oltre che ai chilometri anche il consumo giornaliero di benzina o di altri dati:
- potete realizzare delle tabelle di dati Alfanumerici e questo senza nessuna limitazione(salvo per la lunghezza max di 255 caratteri) sulla lunghezza delle singole stringhe.
Questo puo' subito essere applicato alla programmazione di calendari,agende telefoniche o altre applicazioni dove il computer deve avere contemporaneamente presenti in memoria molti dati:
Riprendete l'esempio di pag. 67 e provate a confrontarlo col seguente:

```
4F$=" BENVENUTO TRA I COMMODORIANI"
5 FOR I=1 TO 10:READ B$(I):NEXT
10 INPUT"DIMMI IL TUO NOME";A$
20 FOR I=1 TO 10
30 IF A$ =B$ (I) THEN 110
40 NEXT
80 PRINT "IL TUO NOME NON E' COMPRESO"
85 PRINT" TRA GLI OPERATORI USUALI"
90 PRINT " SEI SICURO DI CONOSCERE IL PROGRAMMA"
95 INPUT"SI NO";B$
100 IF B$%="SI"THEN PRINT"OK ";A$;F$
```

```
105 IFB$="NO"THEN 1000
106 PRINT"RISPOSTA NON CHIARA":GOTO90
110 DATA FULVIO,LUISA,......
1000 REM ISTRUZIONI PROGRAMMA.
```

E' chiaro che se i nomi da controllare fossero molti (diciamo anche solo 100) l'utilizzo di un vettore diviene obbligatorio PER EVITARE 100 confronti logici, quando ne potete scrivere uno solo inserito in un ciclo e farlo eseguire poi le volte necessarie
Vediamo ora una tabella di indirizzi:

```
10 DIM N$(100),V$(100),L$(100),CB$(100),T$(100)S$(100)
20 FOR I=1 TO 100: PRINT I;
30 INPUT "NOME E COGNOME";N$
40 INPUT "VIA O STRADA ";V$
50 INPUT "LOCALITA' O CITTA'";L$
60 INPUT "CODICE DI AVVIAMENTO";C$
70 INPUT "STATO";S$
80 INPUT "TELEFONO";T$
100 NEXT I
```

Con questo piccolo programma potrete caricare in memoria fino a 100 nomi e indirizzi(e poi con le istruzioni del registratore memorizzarli).Fate pero' attenzione all' occupazione di memoria:
quando avete battuto il programma chiedete con PRINT FRE(0) il numero di Bytes di memoria ancora liberi; fatelo fare al programma durante l'inserimento dati in modo da poter memorizzare gli stessi prima di trovarvi davanti ad OUT OF MEMORY.
Se ora volete conoscere i dati di una persona contenuta nella lista ecco una seconda parte di programma:

```
200 INPUT"NOME DA CERCARE";B$
210 FORI=1TO100:IF B$=N$(I)THEN PRINT"TROVATO":GOTO250
220 NEXT:PRINT"IL NOME NON E' IN ELENCO":GOTO250
250 PRINT V$(I):PRINTL$(I):PRINTC$(I):PRINT"TEL.";T$(I)
260 PRINT S$(I):GOTO 200
```

Provate ora come esempio a programmare una fase per la modifica di un dato, e generate un programma principale che richiami le tre fasi (MENU') d'inserimento modifica e ricerca come se fossero sottoprogrammi

FUNZIONI DEL BASIC

Si passano ora in rassegna alcune altre funzioni non ancora introdotte, ma ugualmente importanti.

Per chiarezza e' bene ricordare la seguente simbologia:

<ESPRESSIONE> e' una parte decisa dall'operatore, necessaria per il completamento della funzione (es. A*B/C)
[ESPRESSIONE] e' una parte opzionale

FUNZIONI DI TEMPO TI TI$

Esistono due funzioni una numerica TI ed una alfanumerica TI$.

Il COMMODORE 64 comincia a contare da zero al momento dell'accensione con una unita' pari ad 1/60 di secondo, percio' se volete conoscere il tempo trascorso dall'accensione chiedete la stampa di TI/60

```
5 PRINT"<HOME>";TI:GOTO5:REM PREMERE IL TASTO CLR/HOME
```

Provate a vedere quanto tempo viene utilizzato da un ciclo FOR NEXT di 5000 cicli

```
5 A = TI
10 FOR I = 1 TO 5000:NEXT
20 B=TI-A
30 PRINT B/60
```

Con questo metodo potete anche confrontare quanto tempo viene impiegato con l' uso di una istruzione basi al posto di un' altra simile:

```
10 A=5 :B=0: C=TI
20 IF A=6 OR A>6 THEN B=B+1
30 IF B=1000 THEN PRINT (TI-C)/60:END
40 GOTO 20
```

oppure:

```
10 A=5 :B=0: C=TI
20 IF A=6 THEN 1000
21 IF A>6 THEN B=B+1
30 IF B=1000 THEN PRINT (TI-C)/60:END
40 GOTO 20
```

potrete cosi' vedere quale e' la differenza di tempo tra l' esecuzione di 1000 IF logici singoli o doppi.
La seconda funzione, TI$, e' alfanumerica e deve essere azzerata prima dell' uso. E' composta da ore, minuti, secondi, percio' vi puo'servire da orologio:

```
10 TI$="000000"
20 PRINT"<CTRL>9"TAB(34)TI$:PRINT"<HOME>";:GOTO 20
```

Avete notato la linea 10? : e' necessaria per azzerare il cronometro. Fate attenzione che per operare su TI$ dovrete trasformare il contenuto in numeri altrimenti incorrete in un errore (si puo' trasformare una stringa in un numero, basta usare la funzione VAL, che incontrerete piu' avanti).

FUNZIONI PER IL TRATTAMENTO DELLE STRINGHE

ASC (STRINGA)
preleva il primo carattere della stringa e ritorna un numero compreso da 0 a 255 che per convenzione e' legato al carattere stesso (questa convenzione e' chiamata ASCII COMMODORE ed e' visibile nelle tabelle registrate in appendice.
Eseguire un ASC di una stringa che non contiene caratteri provoca errore ?ILLEGAL QUANTITY ERROR.

```
10 PRINT ASC("COMMODORE 64")
20 PRINT ASC(S$)
```

CHR$ (NUMERO)
ritorna il carattere che in codice ASCII COMMODORE corrisponde alla variabile numerica assegnata ed e' quindi la funzione inversa di ASC.

```
PRINT ASC (CHR$ (35))
```

L'utilita' di CHR$ si verifica soprattutto nei programmi, infatti alcuni dei 256 numeri (da 0 a 255) provocano non solo caratteri ma anche operazioni di tabulazione, ad esempio:

```
CHR$ (14)  passa al set caratteri minuscolo/maiuscolo
CHR$ (142) passa al set caratteri maiuscolo/grafico
CHR$ (147) equivalente all' HOME
CHR$ ( 17) CRSR GIU'
```

```
CHR$ (145) CRSR SU
CHR$ (157) CRSR SINISTRA
CHR$ ( 29) CRSR DESTRA
CHR$ ( 13) RETURN
CHR$ ( 32) SPAZIO BIANCO
CHR$ ( 18) REVERSE ACCESO
```

anche i colori vengono modificati dalla funzione CHR$

```
5   BIANCO
28  ROSSO
30  VERDE
31  BLU
144 NERO
156 PORPORA
158 GIALLO
159 CYAN
```

Vediamo un esempio:

```
10 PRINT CHR$(147): REM PULISCE LO SCHERMO
15 READ I
20 PRINT CHR$(18) CHR$(I)"**********":REM REVERSE ACCESO
30 REM DEFINIZIONE COLORE E STAMPA DI 10 SPAZI BIANCHI
40 GOTO 15
50 DATA 5,28,30,31,144,155,158,159
```

GET (LISTA DI VARIABILI)
Come gia' detto preleva il primo carattere disponibile dal Buffer di tastiera:

```
10 Q$=""
20 GETQ$:IFQ$=""THEN20:REM ATTENDE LA PRESSIONE TASTO
30 PRINT"HAI PREMUTO ";Q$
```

LEN (STRINGA)
Ritorna la lunghezza di una stringa ed e' molto utile nelle impaginazioni per calcolare i tabulatori.

Centriamo alcune stringhe sul video:

```
10 INPUT "TRE FRASI MINORI DI 40 CARATTERI";A$;B$;C$
20 T=(40-LEN(A$))/2:PRINT TAB(T)A$
30 T=(40-LEN(B$))/2:PRINT TAB(T)B$
40 T=(40-LEN(C$))/2:PRINT TAB(T)C$
```

LEFT$ MID$ RIGHT$

La loro forma e':

```
LEFT$ ( STRINGA>,<INTERO )
MID$  ( STRINGA>,<ESPR.NUM.1>,<ESPR.NUM.2>
RIGHT$ ( STRINGA>,<ESPR.NUM. )
```

LEFT$ preleva una parte dalla stringa contando da sinistra un N. di caratteri pari all'intero scritto (LEFT = SINISTRA).
MID$ preleva dalla stringa una sottostringa prendendo come carattere d' inizio quello indicato dalla prima espressione numerica e contandone tanti quanti vengono indicati dalla seconda espressione.
RIGHT$ preleva una parte dalla stringa contando da destra un N. di caratteri pari all'espressione numerica.

```
10 A$ = "GENNAIOFEBBRAIOMARZO"
20 REM   12345678901234567890
30 REM   98765432109876543210
40 PRINT LEFT$ (A$,7)
50 PRINT MID$ (A$,8,7)
60 PRINT RIGHT$ (A$,5)
```

Notate che con MID$ si possono ottenere anche le funzioni LEFT$ e RIGHT$:

```
70 PRINT MID$(A$,1,7)
80 PRINT MID$(A$,15,5)
```

Con queste funzioni potete modificare la parte terminale di una parola:

```
10 A$=" E' SIMPATICO"
20 INPUT "NOME";B$
30 INPUT "E' DONNA";C$
40 IF LEFT$(C$,1)="S"THEN60
50 PRINT B$;A$:GOTO10
60 A$ = LEFT$(A$,12)+"A":GOTO50
```

oppure incolonnare frasi dopo averle rese tutte lunghe uguali con l' addizione di spazi bianchi:

```
5 B$="                    ":REM 20 SPAZI BIANCHI
10 DIM A(12):FOR I= 1 TO 12
20 READ A$(I):A$(I)=A$(I)+B$
30 A$(I) = LEFT$(A$(I),20):NEXT
40 FOR I= 1TO12
50 PRINT A$(I),
60 NEXT
```

Ulteriori specifiche:

LEFT$(A$,I) con I =0 restituisce una stringa nulla

RIGHT$(A$,I) con I maggiore della lunghezza della stringa restituisce l' intera stringa

MID$(A$,I,J) con I maggiore della lunghezza stringa e J=0 restituisce una stringa nulla

POS
Ritorna il N. di colonna in cui si trova il cursore in quell'istante:

```
10 IF POS = 35 THEN PRINT
```

STR$ (ESPRESSIONE NUMERICA)
trasforma in stringa la rappresentazione numerica calcolata nell'espressione aggiungendo uno spazio a destra.
A sinistra si aggiunge uno spazio solo se il valore e' positivo (viceversa c'e' ma e' occupato dalla lineetta del meno "-").
Se avete bisogno di tabellare dei valori con STR$ potrete trasformare i numeri in stringhe ed operare su di essi le tabulazioni necessarie per incolonnarli.

```
10 INPUT "GIORN";G
20 INPUT "MESE";M
30 INPUT "ANNO";A
40 PRINT STR$(G);"/";STR(M);"/";STR$(A-1900)
```

VAL (STRINGA)

trasforma una stringa, composta di numeri, nel valore numerico da essa rappresentato.
Prende in considerazione solamente i numeri, il punto, i segni + - e la lettera E (per l' esponente in virgola mobile), pertanto se il primo segno fosse diverso da queste lettere il valore sara' reso nullo.

FUNZIONI MATEMATICHE

ABS (ESPRESSIONE)

Restituisce il valore numerico dell'espressione privata

del segno algebrico
(e pertanto sempre positiva):

```
10 A SQR( ABS(-10))
10 B= LOG(ABS(SIN (X/L)))
```

ATN (NUMERO)

Fornisce in radianti l'arco compreso tra -TT/2 e + TT/2 il cui valore di tangente e' pari al numero dato.

```
10 INPUT"H,L";H,L:G=H/L
20 PRINT"PENDENZA DEL PENDIO ";G*100;"%"
30 PRINT"L'ANGOLO OPPOSTO AD H E' DI ";
40 PRINT ATN(G)*200/TT;"GRADI CENTESIMALI"
```

COS (ESPRESSIONE NUMERICA)

Fornisce il valore della funzione COSENO di un arco assegnato in radianti:

```
10 INPUT"ANGOLO CENTESIMALE";A
20 PRINT" IL COSENO VALE ";COS (A*TT/200)
```

SIN (ESPRESSIONE NUMERICA)

Fornisce il valore naturale della funzione SENO di un arco assegnato in radianti:

```
10 INPUT"ANGOLO CENTESIMALE";B
20 PRINT"IL SENO DI";B;" VALE";SIN(A*TT/200)
```

TAN (ESPRESSIONE NUMERICA)

Fornisce il valore naturale della funzione tangente dell' arco espresso in radianti risultante dall'espressione numerica.
La funzione tangente non e' definita per <ESPRESSIONE NUMERICA> molto prossima a TT/2 ed ai suoi multipli negativi o positivi, pertanto dovendola usare e' bene cautelarsi contro interruzioni del COMMODORE 64 a causa di errore (viene segnalato un DIVISION BY ZERO dovuto al quoziente TAN = SIN/COS dove COS(TT/2)=0

Provate il programma:

```
10 FOR I= 1 TO 99
20 A = TT/2 - 10 ↑ (-I):PRINTA,TAN(A)
30 FOR H= 1 TO 500: NEXT H:NEXT I
```

EXP (ESPRESSIONE NUMERICA)

Restituisce il valore della potenza del numero e (2.71828283..) cioe':
A=EXP(5) significa A=e↑5

Poiche' l'elevamento a potenza provoca risultati molto elevati si avra' errore di OVERFLOW per valori dell' espressione numerica superiori a 8.0296919

```
10 SHX = (EXP(X)-EXP(-X))/2
```

LOG (ESPRESSIONE NUMERICA)

Restituisce il valore del logaritmo naturale dell'espressione numerica.
Il logaritmo non e' definito se l'espressione e' <= 0 percio' in tal caso si ha segnalazione di errore? ILLEGAL QUANTITY.

```
10 PRINT LOG (5*2)
```

INT (ESPRESSIONE NUMERICA)

Ritorna la parte intera dell' espressione numerica.
Provate questo programma per capire cosa succede con i numeri negativi e con gli interi:

```
10 PRINT INT(5.26*3.14)
20 PRINT INT(5.26*(-3.14)
30 I% = 5.26*3.14 :PRINT I%
```

RND (ESPRESSIONE NUMERICA)

Restituisce un numero casuale compreso tra 0 e 1.
Per ottenere cio' il COMMODORE 64 esegue dei calcoli partendo da un numero detto "SEME".
Il valore dell'espressione numerica e' utilizzato per il suo segno e solo quando negativo, anche come seme.
In particolare se l'espressione:

- e' positiva si ottengono numeri pseudocasuali partendo da un seme generato all'accensione del sistema.
- se e' nullo si ottengono numeri pseudocasuali direttamente dal clock.
- se e' negativo per ogni seme si ottiene uno stesso ed unico numero casuale :

Provate i seguenti programmi facendoli girare ognuno piu' volte:

```
10 FOR H= 1 TO 10
20 PRINT RND(1);
30 NEXT H
```

oppure lo stesso sostituendo la riga 20 con le seguenti:

```
20 PRINT RND(0)

20 PRINT RND(-1)

20 PRINT RND(-H)
```

Per trasformare i numeri casuali appartenenti all'intervallo 0-1 ad un intervallo piu' ampio dove i valori limite sono A e B la formula e' semplice:

```
X= A + RND ( 0)* (B-A)
```

Se pero' volete che X sia intero,poiche'1 non fa parte della serie di numeri che viene restituita dovrete modificare la formula in:

```
X% = A% + RND (1)*( B%-A%+1)
```

SGN (espressione numerica)

Restituisce, in base al valore dell'espressione numerica, i seguenti valori:

 1 se <ESPRESSIONE NUMERICA> e' positivo
 0 se <ESPRESSIONE NUMERICA> e' nulla
-1 se <ESPRESSIONE NUMERICA> e' negativa

Puo' essere utile per eseguire scelte con l'istruzione ON GOTO (ricordando che quest'ultima richiede valori solo positivi):

```
10 FOR I = 0 TO 2π STEP π/10
15 A= SGN(SIN(I))
20 ON A+2 GOTO 100,200,300
100 PRINTTAB(13+11*SIN(I))"SIN E' NEGATIVO":NEXT
110 END
200 PRINTTAB(13+11*SIN(I))"SIN   E'  NULLO":NEXT
210 END
300 PRINTTAB(13+11*SIN(I))"SIN E' POSITIVO":NEXT
```

SQR (espressone numerica)

Restituisce la radice quadrata positiva dell'espressione numerica.
Se l'espressione e' negativa si ha segnalazione di errore ?ILLEGAL QUANTITY ERROR.
Ricordate anche che A=SQR(B) e' equivalente ad A=B↑(1/2) pertanto potrete estrarre qualsiasi radice usando semplicemente l'elevamento a potenza:

```
10 B=9:A=SQR (B):PRINT A
20 PRINT B↑(1/2)
30 PRINT 64↑(1/3)
40 PRINT SQR(-B)
```

DEF FN < nome > (variabile) = <espressione numerica>

Si verifica spesso che sia necessario calcolare piu' volte od in piu' punti del programma, il valore di una variabile attraverso la stessa espressione numerica.
Nei casi in cui la variabile da definire e' unica si puo' ricorrere alla definizione di una funzione (invece che al GOSUB) avente come nome due sole lettere seguite in parentesi dalla variabile da definire.
L'espressionbe sara' una qualsiasi formula matematica (attenzione matematica e non per stringhe, perche' si avrebbe segnalazione di errore ?TYPE MISMATCH ERROR).
Fate attenzione a porre la definizione prima della chiamata che avviene con l'espressione:

FN < nome > (numero)

Non e' detto che il numero serva sempre (anche se ci deve essere), infatti la funzione potrebbe anche non contenere come argomento la variabile da definire:

```
5 Z=10: S=30 :X=100
10 DEF FN A(X) = X*3+X
20 DEF FN B(X) = 5*100
30 DEF FN C(X) = Z-S
40 REM USO DI FN
50  PRINT  FN A (5) : REM E' L' UNICA LEGATA AL VALORE DI
X
60 PRINT FN B (0) : REM OTTERRETE SEMPRE 500
70 PRINT FN C (0) :REM LO 0 NON SERVE MA CI DEVE ESSERE
```

Non e' possibile usare la DEF FN in modo diretto, con una eccezione: se avete finito di usare un programma nel

quale e' stata definita una funzione, potrete continuare a ch'amare quella funzione gia' definita anche in modo diretto:

```
NEW
10 DEF FN A(X)= X*X-X:END
RUN

PRINT FNA(5)
```

Continuate pure a chiedere FN A con altri valori, poi battete NEW: alla successiva richiesta di FN A avrete errore per mancanza di definizione? UNDEF'D FUNCTION ERROR (causato dal NEW che ha cancellato il programma i dati e la definizione della funzione).

FUNZIONI LOGICHE (LOGICHE?)

Non e' possibile in questa sede dare un'ampia definizione dell'algebra Booleana, percio' per un approfondito apprendimento della materia si rimanda alla bibliografia specifica; di seguito si sono riportate le cose fondamentali che possono servirvi, nella speranza che le definizioni riportate non vi portino davvero a dubitare della logicita' delle funzioni logiche
Il COMMODORE 64 memorizza i dati con una serie di circuiti che funzionano grossolanamente come tanti interrutori in grado di assumere due sole posizioni:

ACCESO 1
SPENTO 0

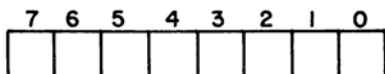
ognuno di questi interrutori viene chiamato:

BIT (si legge bit)

se ora ne mettete 8 in successione l'uno all'altro ottenete un:

BYTE (si legge baite)

ossia una striscia di otto caselle in cui puo' apparire un 1 od uno 0.

7	6	5	4	3	2	1	0

Voi avete imparato a contare utilizzando 10 numeri:
0,1,2,3,4,5,6,7,8,9

e quando li avete esauriti ottenete il numero successivo formandolo con due cifre, la prima e' 1 la seconda varia ancora da 0 a 9, poi la prima diviene 2 e cosi' via fino a 99. Ora, esaurite tutte le possibili combinazioni con due numeri passate a tre: quello a sinistra e' il piu' importante, quello a destra il meno: 100,101,.....999. Se l' uomo avesse solamente due dita invece che dieci, probabilmente il modo di contare usuale sarebbe proprio quello Binario, cioe': 0,1 e poi esauriti i numeri a disposizione, si continuerebbe con un numero di due cifre: 10, e poi 11, cosi' esaurite tutte le combinazioni possibili con due cifre si passerebbe a tre: 100, 101, 110, 111: cosi' fa il vostro calcolatore utilizzando le sue due dita, cioe' il Bit.
I numeri interi che vengono introdotti da tastiera con numerazione decimale, vengono trasformati in numerazione fatta da soli 1 e 0, cioe' " BINARIA "

n.dec.	byte	n.dec.	byte	n.dec.	byte
0	0	8	1000	16	...10000
1	1	9	1001	17	...10001
2	10	10	1010	18	...10010
3	11	11	1011	19	...10011
4	100	12	1100	32	..100000
5	101	13	1101	64	.1000000
6	110	14	1110	128	10000000
7	111	15	1111	255	11111111

Potete notare comne esista una relazione tra le potenze di 2 ed i numeri binari, infatti tutti i numeri binari aventi la prima cifra uguale a 1 e le successive a 0 corrisponde in decimale alla potenza di due avente esponente pari al numero degli zeri:

```
 2 (dec) = 10 (bin)
 4 (dec) = 2 * 2 = 100 (bin)
 8 (dec) = 2 * 2 * 2 = 1000 (bin)
16 (dec) = 2 ↑ 4 = 10000 (bin)
32 (dec) = 2 ↑ 5 = 100000 (bin)
```

percio' continuando:

```
64 = 2 ↑ 6    128 = 2 ↑ 7    256 = 2 ↑ 8
```

attenzione se 256 (dec) = 100000000 (8 zeri) (bin)
il numero prima 255 (dec) = 11111111 (bin)

e' il massimo numero che puo' essere rappresentato con un Byte.
Con due Byte potete rappresentare il numero 65535, ma il vostro COMMODORE 64 sacrifica il primo Bit del primo Byte per il segno (0 = + 1 = −) e pertanto il massimo intero rappresentabile con due Byte (come e' gia' stato detto nel paragrafo delle variabili intere) va da -32768 a +32767.

Se ora volete trasformare un numero (es:15793) decimale in binario potrete procedere cosi':

15793 : 2=7896	resto	1	(vale 1)
7896 : 2=3948	resto	0	(vale 2)
3948 : 2=1974	resto	0	(vale 4)
1974 : 2= 987	resto	0	(vale 8)
987 : 2= 493	resto	1	(vale 16)
493 : 2= 243	resto	1	(vale 32)
246 : 2= 123	resto	0	(vale 64)
123 : 2= 61	resto	1	(vale 128)
61 : 2= 30	resto	1	(vale 256)
30 : 2= 15	resto	0	(vale 512)
15 : 2= 7	resto	1	(vale 1024)
7 : 2= 3	resto	1	(vale 2048)
3 : 2= 1	resto	1	(vale 4096)
1 : 2= 0	resto	1	(vale 8192)

ed ora il numero di binario sara' dato dai resti ordinando l'ultimo in posizione piu' elevata e via via fino al primo in posizione piu' bassa.

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
0	0	1	1	1	1	0	1	1	0	1	1	0	0	0	1
32768	16384	8192	4096	2048	1024	512	256	128	64	32	16	8	4	2	1

Questo numero puo' essere ritrasformato in decimale, basta evidenziare solo i valori delle potenze di due delle caselle contenenti 1 e sommarli.

8192+4096+2048+1024+256+128+32+16+1 = 15793

Prima di proseguire provate ancora ad eseguire degli esempi, trasformate in binario i numeri decimali 1023, 32767, 32768, 65535.

Sui numeri binari possono essere eseguite le seguenti operazioni logiche:

< esprssione > AND < espressione >

Opera su due binari e restituisce, per ogni Bit:
1 nel caso che i due numeri siano entrambe 1,
0 se uno dei due o entrambe sono 0:

```
1 AND 1=1          1 AND 0=0
0 AND 1=0          0 AND 0=0
```

Se i numeri binari sono piu' grandi l'operazione viene eseguita per coppie di Bit corrispondenti:

```
 1101 AND 1011  si esegue cosi':

 1101
 1011
======
 1001
```

Da tastiera e da programma l'operatore AND accetta numeri decimali, li trasforma in binario, opera l'AND, li ritrasforma in decimale.

```
100 PRINT 10 AND 7

(dec) 10 = 1010 (bin)
(dec)  7 = 0111 (bin)
           ======
10 AND 7 = 0010 (bin) = 2 (dec)
```

Poiche' sul COMMODORE 64 i numeri interi sono rappresentati con 2 bytes non e' possibile richiedere un AND per numeri superiori a 32767 o inferiore a -32768.
Eseguire l'AND di due numeri ha poco senso tranne quando volete leggere il bit di un byte.
Se ad esempio desiderate conoscere il contenuto del quinto bit del byte di una certa locazione di memoria potete eseguire l'AND di quel byte con il numero binario 10000 (16 decimale) cioe' eseguire PRINT PEEK (loc di mem) AND 16 ed il risultato sara' 10000 ossia 16 solo se il quinto bit contiene 1, viceversa sara' 0.
Per l'uso di AND nei confronti logici, vedere quanto detto nelle espressioni
IF THEN.

< espressione > OR < espressione >

Opera su due numeri (compresi tra -32768 e +32767) li

trasforma in binari e poi su ogni coppia corrispondente opera un calcolo generando:
0 solo se entrambe sono nulli;

1 negli altri casi
poi ritrasforma il numero ottenuto in decimali:

```
0 OR 0 = 0     0 OR 1 = 1
1 OR 0 = 0     1 OR 1 = 1
```

12 OR 6 viene cosi' eseguito:

```
(dec) 12 = 1100 (bin)
(dec)  6 =  110 (bin)
          ======
12 OR  6 = 1110 (bin) = 14 (dec)
```

Per l'uso di OR nei confronti logici vedere quanto detto nella espressione IF THEN.
Le istruzioni AND e OR sono utili usando una POKE per poter modificare una parte di un byte.
Sia dato ad esempio il byte:

per azzerare il terzo e il quarto bit si usera' una maschera come la seguente: 11110011 = 243 (dec)

157 AND 243 = 145 ossia:

pertanto si eseguira':

POKE <loc di mem>, PEEK (loc di mem) AND 243

Per portare invece ad 1 il sesto ed il secondo Bit si usera' una maschera come la seguente:

0	0	1	0	0	0	1	0

157 OR 34 = 191 (dec) ossia:

1	0	1	1	1	1	1	1

pertanto si eseguira':

POKE <loc di mem>, PEEK (loc di mem) OR 34

NOT < espressione >

E' utile in binario quando si devono eseguire operazioni di differenza tra numeri binari o cambiare di segno ad un binario.
Infatti in tal caso il numero da sottrarre viene trasformato in un numero detto "complemento a 2" proprio con l'operazione logica NOT.
Se il numero e' positivo viene cambiato di segno ed il suo valore assoluto viene aumentato di 1

NOT 6=-7
NOT 1=-2
NOT 0=-1

Se negativo diviene positivo ed il suo valore assoluto diminuisce di 1

NOT (-5) = 4

Se il numero e' reale viene trasformato in intero e se e' esterno ai valori (- 32768 + 32767) si ha la segnalazione? ILLEGAL QUANTITY ERROR.
NOT e' anche usato per esprimere il contrario in una relazione di confronto:

```
10 IF NOT (A=F) THEN 50
20 PRINT" A E' UGUALE A F"
30 END
50 PRINT" A E' DIVERSO DA F"
```

ISTRUZIONI DI COMANDO SISTEMA:

CLR

Provoca la perdita di tutti i dati memorizzati (salva i programmi, che continuano ad essere presenti), piu' precisamente si eliminano:

- le variabili ed i loro valori
- le tabelle
- gli indirizzi memorizzati per i GOSUB ed i FOR NEXT
- le funzioni inserite con DEF FN

la memoria prima occupata diviene disponibile per qualsiasi altra memorizzazione.
FATE ATTENZIONE non usate CLR se state registrando su altre periferiche, in quanto perdereste i dati di quei " FILES " che non sono stati ancora correttamente chiusi:

```
5 DIM A (5000)
10 FOR I = 1 TO 5000
20 A (I) = I : NEXT
30 PRINT FRE  (0): REM MEMORIA ANCORA DISPONIBILE
40 CLR
50 PRINT FRE  (0): REM MEMORIA DISPONIBILE DOPO IL CLR
```

Date RUN dopo alcuni secondi ottenete:

```
 13823 pari alla memoria ancora disponibile dopo averne
       occupata buona parte con il vettore A (I)
-26694 legato alla memoria disponibile dopo aver
       "ucciso" il vettore A (I) in questo modo:
```

65536 - 26694 = 38842 = MEMORIA DISPONIBILE

FRE (variabile)

Restituisce la memoria disponibile in quel momento ed utilizzabile per programmi e o variabili.
Quando il valore dovesse essere negativo aggiungete 65536 ed otterrete il dato voluto.
Se FRE (V) = 0 il calcolatore non ha piu' memoria e segnala un OUT OF MEMORY.

La variabile o il numero tra parentesi non hanno nessuna influenza (e non chiedetevi allora perche' ce ne' bisogno...!)

Ricordate che un numero intero occupa 2 byte mentre un numero reale occupa 5 byte:

```
10 PRINT FRE (5)
100 IF FRE (1)<500 THEN PRINT"POCA MEMORIA DISPONIBILE"
```

CONT

provoca la CONTinuazione di un programma che si e' arrestato con uno STOP un END o col tasto RUN STOP.
Il programma continua dal punto in cui era avvenuto l'arresto salvo che:
- si sia modificata o riconfermata con RETURN una qualche linea programma
- siano avvenuti errori nel programma che ne abbiano provocato l'arresto.

OPEN < NC >, < NP >, < IS >, " NF,T,M "

E' una delle istruzioni piu' complicate da "digerire", ma, per fortuna, viene usata solamente per colloquiare con periferiche quali il registratore nel caso di files sequenziali, la stampante, oppure il drive, ed in alcuni casi non e' neppure necessario che vengano specificate tutte le parti dell'istruzione.

NC = Numero associato al file aperto,	
NP = Numero della periferica	0 = tastiera
	1 = registratore
	3 = video
	4 = stampante
	8 = unita' disco
IS = Indirizzo secondario	
NF = Nome del file	
T = Tipo del file	P = programma
	S = sequenziale
	R = relativi
	U = utilizzatore
M = Modo di apertura del file	R = lettura
	W = scrittura.

Con la OPEN si apre un "canale" attraverso il quale si provvede a trasferire i dati in entrata od in uscita tra la periferica e l'unita' centrale.
Per non avere confusione e' bene prendere in esame la OPEN nei confronti di ogni singola periferica.

REGISTRATORE A CASSETTE

NC va da 1 a 127 deve essere sempre presente
NP vale 1 e puo' essere anche omesso
IS serve ad avvertire la periferica di eseguire:
   0 operazioni di lettura
   1 operazioni di scrittura con chiusura del file
   2 operazioni di scrittura con chiusura del file
     con scrittura di fine nastro.
NF nome del file, puo' essere anche omesso, ma diviene
   difficoltosa la ricerca del programma; non deve
   avere piu' di 16 caratteri.
T tipo di file   NON SI USA
M modo di apertura NON SI USA.

Il registratore e' una periferica "poco" intelligente: potrete senza segnalazione alcuna registrare dati su altri files gia' esistenti (con la evidente perdita dei primi), pertanto prima di effettuare operazioni di scrittura e' sempre bene controllare che cosa contiene la cassetta.
Vediamo ad esempio come eseguire la memorizzazione (salvataggio) di un programma:

```
SAVE "nome programma"
```

oppure:

```
OPEN 5,1,1   (od anche OPEN 5,1,2)
SAVE "nome programma",1,1
```

Registrazione di un file dati sequenziale:

```
10 OPEN 4,1,1,"NOME FILE"
20 FOR I=1 TO 20: PRINT#4,I:NEXT I
30 CLOSE 4
```

Lettura di un file dati:

```
100 OPEN 5,1,0, "NOME FILE"
120 REM OPPURE OPEN 5,1,"NOME FILE"
130 FOR I=1 TO 20:INPUT#5,A:PRINT A:NEXT I
140 CLOSE 5
```

STAMPANTE:

NC = da 1 a 127
NP = 4

```
IS = dipende dalla stampante ed e' utile per dare ordini
     alla stampante (0 = stampa maiuscolo-grafica
                     7 = stampa minuscolo-maiuscolo etc)
"NF,T,M" non vengono usati.

10 OPEN 6,4: PRINT#6,"STAMPA CON LA STAMPANTE"
20 CLOSE 6

30 OPEN 5,4,0:PRINT#5,"STAMPA MAIUSCOLO":CLOSE 5
```

DRIVE

```
NC = da 1 a 127
NP = 8 (nel caso di piu' registratori da 8 a 15)
IS = da 2 a 14 per comunicazioni normali.
     0 ed 1 sono usati dal sistema per operazioni di
     lettura e scrittura;
     15 serve per passare comandi al disco, leggere
     errori, formattare il disco e altro.

NF = come per il registratore e' il nome del file del
     programma e non puo' superare i 16 caratteri.
T  = sul disco si possono registrare i seguenti tipi
     di files:
     P = programmi
     S = files sequenziali
     R = files relativi
     U = files di comandi
M  = i modi sono sempre:
     R per lettura (se tralasciate M il file e'
       aperto in lettura);
     W per scrittura (il disco non deve avere
       la protezione).
```

Il drive e' in grado di controllare se sta' scrivendo un file di nome uguale ad uno esistente, in tal caso si ha segnalazione di errore ? FILE EXIST (e non viene registrato). Se proprio volete scrivere su di un file esistente (col rischio che un attimo di mancanza di energia vi faccia perdere entrambi i files, dovete far precedere il nome del file dalle seguenti tre lettere @0: cio' puo' avvenire anche in vari modi: scriviamo ad esempio sul file "DATI,S" esistente:

Primo modo:
```
OPEN 2,8,2,"@0:DATI,S,W"
```

secondo modo:
```
OPEN 2,8,2,"@0:"+"DATI"+",S,W"
```

terzo modo:

```
A$="@0:"
B$="DATI"
C$=",S,W"
D$=",S,R"
OPEN 2,8,2,B$+C$:   REM SCRITTURA DI NUOVO FILE
OPEN 2,8,2,A$+B$+C$:REM SCRITTURA SU FILE GIA' ESISTENTE
OPEN 2,8,2,B$+D$:   REM LETTURA DEL FILE
```

E' errato aprire un file gia' aperto perche' si ha segnalazione di errore: ? FILE OPEN ERROR; in tal caso la segnalazione di errore chiude anche il canale aperto ossia "aprire un canale aperto ne provoca la sua chiusura".

```
10 OPEN 15,8,15:PRINT#15,"NEW 0,NOME,ID"
20 CLOSE 15
```

```
10 OPEN 7,8,7,"DATI,S,W"
20 FOR H=1 TO 100 PRINT#7,H:NEXT H
30 CLOSE 7
40 OPEN 6,8,6,"DATI,S"
50 FOR I=1 TO 100:INPUT#6,A:PRINT A:NEXT I
60 CLOSE 6
```

Altre informazioni verranno date nell'esaminare le successive istruzioni e nei manuali delle rispettive periferiche.

CMD < NC >, [<stringa>]

trasferisce il comando di visualizzazione dal video alla periferica collegata tramite il canale NC, con formato di stampa identico a quello del video; la trasmissione dei dati avviene sul canale NC, sempreche' in precedenza sia stata eseguita la OPEN relativa. Il collegamento e' valido fino a che viene eseguito un PRINT#NC:questo comando blocca l'attesa di ulteriori informazioni da parte della periferica, il canale pero' rimane aperto e potete ancora passare dati alla periferica con un altro CMD, oppure con l'istruzione PRINT#; in ogni modo dopo un CMD e' sempre necessario eseguire un PRINT# e successivamente un CLOSE.
Dimenticando il PRINT# si avra' un cattivo funzionamento del video e la periferica non sara' stata chiusa correttamente.
La <STRINGA> che appare quale seconda opzione nel comando, viene stampata come se aveste dato un ordine di PRINT e puo' essere utile per dare un titolo alle stampe successive.

```
OPEN 1,4:CMD 1,TITOLO DEL PROGRAMMA:LIST
```

vi permettera' di listare il programma; ultimata la stampa battere:

```
PRINT#1:CLOSE 1
```

```
10 OPEN 1,1,2,"PROVA,S,W":CMD 1
20 PRINT "PRIMA STRINGA"
30 PRINT "SECONDA STRINGA"
40 PRINT "TERZA STRINGA"
50 PRINT#1,"ULTIMA STRINGA"
60 PRINT "QUESTA VA SUL VIDEO"
70 PRINT#1,"QUESTO ANCORA SUL NASTRO"
80 CLOSE 1
100 OPEN 2,1,"PROVA"
110 INPUT#2, A$,B$,C$,D$:CLOSE 2
120 PRINT A$,B$:PRINT C$,D$
```

Se si verificassero segnalazioni di errore queste verranno sempre trasmesse sullo schermo anche se avete attivato il CMD.

PRINT# <NC>, <VARIABILE>, /;<VARIABILE>...

dove NC e' il numero del canale associato al file nella OPEN eseguita in precedenza. Questa istruzione provoca la scrittura sulla periferica collegata al computer utilizzando il canale aperto dalla OPEN, e' bene tenere separati i due casi di scrittura con stampante e scrittura su dischi o cassette.

- STAMPANTE: valgono le cose viste per l'istruzione PRINT, avrete una stampa ad inizio riga con la prima PRINT mentre le successive saranno regolate dalla presenza del ";" o della "," che avrete posto da ultime alla fine dati della PRINT precedente.

```
OPEN 1,4:PRINT#1,"PRIMA STAMPA","SECONDA";"TERZA":CLOSE1
```

Se volete provocare un ritorno del carrello a capo potete usare anche il carattere CHR$ (13)

```
OPEN 1,4:PRINT#1,"PRIMA STAMPA"CHR$(13)"SECONDA":CLOSE1
```

- DISCHI O NASTRI: la cosa cambia leggermente perche' la virgola, il punto e virgola possono causare

inconvenienti in fase di lettura, i dati vengono scritti continuativamente:

```
10 OPEN 1,1,1,"DATI"
20 FOR I=1 TO 20:PRINT#1,I;:NEXT I
25 CLOSE 1:INPUT "RIAVVOLGERE E PREMERE RETURN";Q$
30 OPEN 1,1,0,"DATI":REM RIAVVOLGERE IL NASTRO
40 FOR I=1 TO 20: INPUT#1,A:PRINT A:NEXT I
50 CLOSE 1
```

```
10 OPEN 2,1,1,"DATI ALFANUM"
20 FOR I=1 TO 5:PRINT#1,"DATO"+STR$(I),:NEXT I
30 CLOSE 2:REM RIAVVOLGERE IL NASTRO
40 OPEN 1,1,0,"DATI ALFANUM"
50 FOR I=1 TO 5:INPUT#1,A$:PRINT A$:NEXT I
```

Si omette il CLOSE1 perche' il programma andra' in errore, infatti avrete letto il primo A$ cosi':

DATO1DATO2DATO3DATO4DATO5

Se volete evitare questo inconveniente avete due possibilita':

- Separate ogni stringa dalla successiva con CHR$ (13)

esempio: PRINT#1,A$,CHR$(13),B$,CHR$(13) etc.

- Usate un PRINT#1 per stampare un solo dato:

esempio:PRINT#1,A$:PRINT#1,B$: etc

evitando virgole, punti e virgola e altri caratteri separatori.
Evitate, se possibile, di memorizzare su supporti magnetici stringhe piu' lunghe di 80 caratteri, perche' avrete poi inconvenienti nel cercare di rileggerle.(vedere INPUT#).

INPUT# < NC > , < lista di variabili >

Provoca il ricevimento di tutti i dati dalla periferica sul canale aperto con la OPEN che specifica il numero del file.
Come gia' detto per INPUT quando il dato atteso come numerico e' invece alfanumerico si ha segnalazione di errore (BAD DATA ERROR).
Come dato viene prelevato tutto quanto e' compreso tra due caratteri separatori (virgola, punto e virgole, CHR$ (13), [RET], etc.). Pero' nel caso di stringhe piu'

lunghe di 80 caratteri si verifica un errore: STRING TOO LONG, per tale evenienza sara' necessario usare l'istruzione GET#.
Tenete infine presente che si puo' anche ricevere il dato dal video (e' anche lui una periferica ed ha come NP=3) in tal caso verra' letta tutta la linea logica corrispondente alla posizione del cursore in quell'istante e successivamente il cursore verra' spostato a capo nella linea successiva:

```
10 OPEN 3,3:PRINT A$
20 PRINT "QUESTO E' IL DATO <SHIFT> <CRSV>"
30 INPUT#3,A$:PRINT A$
40 CLOSE 3
```

Provate a ripetere il programma sovrapponendo alla prima riga in cui e' apparso "QUESTO E' IL DATO" altri caratteri fino alla destra del video, constaterete come venga letta tutta la riga video.
Provate a modificare la riga 20:

```
20 PRINT "QUESTO E' IL DATO, ABC, 123 <SHIFT> <CRSV>"
30 INPUT#3,A$,B$,C:PRINT A$,B$,C
```

Poiche' la virgola e' un carattere separatore, la lettura di A$ ha prelevato solo la prima parte dei dati scritti sul video:

```
20 PRINT "QUESTO E' IL DATO; ABC; 123 <SHIFT> <CRSV>"
```

Otterrete un risultato che non e' strano solo se si pensa che il punto e virgola ha unito le stringhe per cui:

```
A$="QUESTO E' IL DATOABC123"
B$="" e percio' non appare,
C=0 in quanto non e' stato letto
```

E' importante notare che non si ha segnalazione di errore anche se in questo caso due dati sono stati posti a zero.

GET# <NC>, <lista variabili>

L'istruzione preleva sul canale aperto con la OPEN un carattere per volta dalla periferica. Il carattere viene letto anche se e' un carattere di controllo, in particolare se si richiede la lettura di una stringa e la periferica non la fornisce, l'istruzione prende un

stringa nulla "", nel caso di variabile numerica invece viene posta a zero.

```
10 OPEN 5,1,1,"DATI"
20 PRINT#5,20,21;30,40 CHR$13 50,"ABC"
30 CLOSE 5: STOP:REM RIAVVOLGERE IL NASTRO
40 OPEN 6,1,0,"DATI"
45 FOR I=1 TO 13
50 GET#5,A$:PRINT A$:NEXT I:CLOSE 5
```

Nel caso in cui fosse stata registrata una stringa superiore ad 80 caratteri potrete leggerla nel seguente modo:

```
10 OPEN 2,1,0,"DATI":B$=""
20 GET#2,A$
30 IF A$=CHR$(13)THEN 50
40 B$=B$+A$:GOTO 30
50 PRINT B$: CLOSE 2:STOP
```

Quando si legge un carattere col GET# dal video il cursore avanza al carattere successivo.

CLOSE (NC)

Serve per scollegare la periferica dall'unita' centrale relativamente al numero di file aperto con l'istruzione OPEN eseguita in precedenza. E' importante ricordarsi di ordinare l'istruzione CLOSE soprattutto quando si sono registrati dei file dati su dischi o nastri, infatti la chiusura del file provocata con la CLOSE rende possibile la successiva lettura di quel file (se non chiudete il file non lo potrete piu' leggere se e' su disco, mentre il registratore non si arrestera' alla fine del file continuando ad avvolgere il nastro).

SAVE "<NOME DEL FILE>", NP,IS

Serve per memorizzare su nastro o su disco il programma che avete in memoria. Non puo' essere usato per files diversi da quelli programma.
Durante l'esecuzione del SAVE il video "sparisce" ed e' bene non toccare la tastiera. Il nome del programma puo' anche essere omesso, ma e' preferibile, per le ricerche successive, assegnare almeno una sigla (il nome comunque non puo' mai superare i sedici caratteri).
NP e' il numero della periferica su cui salvare il programma, se questo viene omesso il COMMODORE 64

inviera' il programma al registratore a cassette e se questo non ha tasti premuti avrete sul video la segnalazione:

PRESS RECORD & PLAY ON TAPE

IS serve per provocare il salvataggio del programma con alcune particolarita':
1 - ordina il caricamento partendo da una locazione di memoria diversa da 2048 che e' quella standard;
2 - registra a fine programma un segnale di fine nastro;
3 - combina le due funzioni 1 e 2.

```
SAVE "NOME PROGRAMMA" andra' sul nastro
SAVE                  andra' sul nastro ma senza nome
SAVE "PROGR",8        andra' sul disco
100 SAVE "LIN"        salva il programma da programma
```

VERIFY "<NOME FILE>", NP

E' un comando usato solo direttamente per verificare se la registrazione del file sulla periferica NP, e' avvenuta correttamente; bisogna ovviamente riavvolgere il nastro. Anche in tal caso se il numero della periferica NP viene omesso la verifica viene effettuata su registratore.
Se la registrazione non e' avvenuta correttamente si ha segnalazione di errore ? VERIFY ERROR.

SYS <locazione di memoria>

Provoca l'esecuzione di un programma in linguaggio macchina (detto ASSEMBLER) gia' presente in memoria ed avente inizio alla <locazione di memoria> indicata in numerazione decimale. Quando tale programma e' ultimato, e se l'ultima istruzione e' un ordine di ritorno (equivalente al RETURN del Basic) si ritorna all'elaborazione in Basic dall'istruzione successiva alla SYS.
Questa istruzione si puo' comandare in modo diretto o da programma.

USER (espressione numerica)

Provoca l'esecuzione di un sottoprogramma in Assembler col ritorno di un dato; si usa in questo modo:
- una parte del programma Basic carica con la funzione POKE la parte di linguaggio macchina relativa al sottoprogramma a partire da una locazione di memoria L
- la locazione L va inserita nelle posizioni di memoria 785,786 usando sempre l'istruzione POKE, per fare cio' L va espresso in Binario (16 bit) i primi 8 devono essere ritrasformati in un numero decimale, i secondi pure, infine si posiziona il primo numero decimale nella locazione 786 ed il secondo in 785;
- quando e' necessario richiamare l'esecuzione del sottoprogramma si ordina USER (espressione numerica) dove il risultato dell'<espressione numerica> viene memorizzata dalla locazione 97, da cui verra' poi prelevata durante l'esecuzione del programma in Assembler;
- alla fine USER ha il valore del risultato della elaborazione avvenuta in Assembler come se fosse una qualsiasi altra funzione, e come tale potra' essere inserita anche in altre espressioni (IF, ON...GOTO).

Supponiamo ad esempio di memorizzare un sottoprogramma Assembler a partire dalla locazione di memoria 820, poiche' risulta:

(dec) 820 =

0	0	1	0	0	0	0	1	0	0	0	0	0	1	0	0
			HI								LI				

Il primo in decimale vale 51
Il secondo in decimale vale 4

pertanto programmerete POKE 786,51:POKE 785,4

altri esempi:

```
100 B=USR(A+B)
150 ON USR (A) GOTO 200,400
```

USO DEI TASTI F1 - F2 - F8

Sul lato destro del vostro COMMODORE 64 sono presenti quattro tasti che possono essere usati per funzioni per

funzioni (F1, F3, F5, F7 se premuti direttamente, e F2, F4, F6, F8 se premuti con lo <SHIFT).
L'esecuzione delle funzioni ha tuttavia bisogno di essere implementata con la programmazione del Basic; la routine deve prevedere una attesa di caratteri, ed il rinvio al sottoprogramma corrispondente alla funzione.
Per creare direttamente il carattere corrispondente al tasto funzione premete tale tasto dopo aver aperto le ".

```
100 Q$="":REM AZZERAMENTO CARATTERE
110 GET Q$:IF Q$=""THEN 110:REM ATTESA CARATTERE
120 IF Q$="<F1>" THEN 200: REM "PREMERE F1"
130 REM QUI IL PROGRAMMA PROSEGUE
200 REM QUI VIENE ESEGUITO IL SOTTOPROGRAMMA
```

In effetti la stessa cosa potrebbe essere fatta con qualsiasi tasto della tastiera, con le Funzioni diviene piu' facile da eseguire.

```
20 INPUT"DATO DA ELABORARE";A:PRINT"■";
30 PRINT"■■■■■■MENU'"
40 PRINT"■PREMERE ■F1■ PER IL QUADRATO"
50 PRINT"■PREMERE ■F3■ PER LA RADICE QUADRATA"
60 PRINT"■PREMERE ■F5■ PER IL C U B O"
65 PRINT"■PREMERE ■F7■ PER FINIRE "
70 Q$=""
80 GETQ$:IFQ$=""THEN80
90 IF Q$="■" THEN 200:REMPREMERE F1 DOPO LE "
100 IF Q$="■" THEN 300:REMPREMERE F3 DOPO LE "
110 IF Q$="■" THEN 400:REMPREMERE F5 DOPO LE "
120 IF Q$="■" THEN 20:REM PREMERE F7 DOPO LE "
130 IF Q$="■" THEN END
200 PRINT"■■IL QUADRATO DI ";A;"E'";A↑2
210 GOTO500
300 PRINT"■■LA RADICE QUADRATA DI";A;"E'";SQR(A)
310 GOTO500
400 PRINT"■■IL CUBO DI";A;"E'";A↑3
500 FOR I=1TO2000:NEXT:GOTO20
```

I caratteri generati dai tasti F1, F2, ... F8 sono gli stessi in modo Reverse dei seguenti:E,I,F,J,K,H,L e cio' si puo' agevolmente vedere battendo le " seguite dai tasti funzione e passando successivamente al modo caratteri minuscolo-maiuscolo (premete contemporaneamente [SHIFT] e [C=]).

USO DEI JOYSTICK E DELLE PADDLES

Esistono sul fianco del vostro COMMODORE 64 due porte (numerate nella figura 1 con 3 e 4) che vengono utilizzate per ricevere comandi da manopole di due tipi diversi.
I JOYSTICK sono dotati di manopola verticale inclinabile in quattro direzioni perpendicolari tra loro di un pulsante.
Le PADDLES hanno invece una manopola rotante piu' un pulsante.
Mentre i primi si possono utilizzare anche con un programma in linguaggio Basic, le seconde richiedono l'uso preferenziale di un piccolo programma in linguaggio macchina (Assembler).

JOYSTICK: passano dati ai due seguenti registri:

porta A registro di memoria in locazione 65320
porta b registro di memoria in locazione 65321

di questi registri vengono modificati i primi cinque Bit (quelli bassi) nel seguente ordine (supponendo di impugnarlo correttamente):

Bit 0 viene attivato spingendo la manopola in avanti
Bit 1 viene attivato tirando la manopola indietro
Bit 2 viene attivato spostando la manopola a sinistra
Bit 3 viene attivato spostando la manopola a destra
Bit 4 viene attivato dal pulsante

Ogni singolo Bit vale normalmente 1, diviene 0 quando la manopola e' stata spostata nella direzione corrispondente o il pulsante premuto. Successivamente il Bit viene riposizionato ad 1 con un intervallo uguale a quello dato dal COMMODORE 64 alla tastiera per la ricerca dei caratteri premuti.
Per leggere se il JOYSTICK e' stato premuto si usera' quindi la funzione PEEK (56320) oppure PEEK (56321).

Provate col seguente programma dopo aver inserito un JOYSTICK in porta A:

```
10 PRINT PEEK (56320):GOTO 10
```

Muovendo la manopola del JOYSTICK o premendo il pulsante avrete notato una variazione nei numeri stampati sul video dal programma, cio' e' dovuto al fatto che ogni

movimento della manopola ha provocato una variazione del valore contenuto nel registro 56320 e quindi un valore decimale diverso letto dalla funzione PEEK (56320).
Se ora eseguite un AND del valore letto col valore 16 (che in Binario vale 10000) otterrete un dato che vale:

16 se il pulsante e' inattivo
 0 se il pulsante e' stato premuto

Infatti provate a disegnarvi il registro 56320 quando non e' attivato:

Porta 1 dopo aver premuto il pulsante:

Pertanto il comando sara':

```
IF PEEK (56320) AND 16 = 0 THEN 1000
```

Se l'espressione e' nulla il pulsante e' stato premuto e si prosegue l'elaborazione in riga 1000.

Infine eseguite AND 15 per rilevare lo spostamento della manopola:

porta inattiva	0	0	1	1	1	1	1	1
BIN. 15	0	0	0	0	1	1	1	1
porta AND 15	0	0	0	0	1	1	1	1

manopola in avanti (Bit 0 messo a 0):

porta	0	0	1	1	1	1	1	0	
porta AND.15	0	0	0	0	1	1	1	0	= dec 14

manopola indietro (Bit 1 messo a 0):

porta	0	0	1	1	1	1	0	1	
porta AND 15	0	0	0	0	1	1	0	1	= dec 13

manopola a sinistra (Bit 2 messo a 0)

porta	0	0	1	1	0	1	1	1	
porta AND.15	0	0	0	0	0	1	1	1	= dec 7

manopola a destra (Bit 3 messo a 0)

porta	0	0	1	1	1	0	1	1	
porta AND 15	0	0	0	0	1	0	1	1	= dec 11

Poiche' il posizionamento obliquo della manopola portera' ad ulteriori combinazioni (infatti azzerera' i due Bit delle due direzioni corrispondenti), e' utile

sottrarre da 15 il risultato scrivendo l'espressione:

15 - (PEEK (65320) AND 15)

e confrontare il risultato finale con la seguente tabella:

```
 1  MANOPOLA AVANTI
 2  MANOPOLA INDIETRO
 4  MANOPOLA A SINISTRA
 5  MANOPOLA AVANTI-SINISTRA
 6  MANOPOLA INDIETRO-SINISTRA
 8  MANOPOLA A DESTRA
 9  MANOPOLA AVANTI-DESTRA
10  MANOPOLA INDIETRO-DESTRA
15  PULSANTE PREMUTO
```

Tutto quanto sopra e' valido anche per la porta B se si usa il registro 65321 al posto del registro 65320.

PADDLES

Il valore passato dalle PADDLES al calcolatore influenza i registri di memoria in locazione 54297 e 54298 ma per evitare errori e' bene usare il seguente programma che viene riportato integralmente dal manuale della Commodore:

```
10 C=12*4096:REM IMPOSTA IL PUNTO DI PARTENZA SOTTOPROCEDURA
11 REM SCRIVE TRAMITE POKE NELLA SOTTOPROCEDURA DI LETTURA
12 REM DELLE PADDLE
15 SYSC:REM RICHIAMA LA SOTTOPROCEDURA DELLE PADDLE
30 P1=PEEK(C+257):REM IMPOSTA IL VALORE DELLA PADDLE 1
40 P2=PEEK(C+258):REM  "   "      "  "   "        "  2
50 P3=PEEK(C+259):REM  "   "      "  "   "        "  3
60 P4=PEEK(C+260):REM  "   "      "  "   "        "  4
61 REM LEGGE LO STATO DEL PULSANTE DI SPARO
62 S1=PEEK(C+261):S2=PEEK(C+262)
70 PRINTP1,P2,P3,P4:REM STAMPA I VALORI DELLE PADDLE
72 REM STAMPA LÒ STATO DEL PULSANTE DI SPARO
75 PRINT:PRINT"FIRE A ";S1;"FIRE B ";S2
80 FORW=1TO50:NEXT:REM BREVE ATTESA
           SHIFT CLR HOME
90 PRINT"♥" :PRINT:GOTO20:REM AZZERA LO SCHERMO E RICOMINCIA
95 REM DATI PER SOTTOPROCEDURA IN LINGUAGGIO MACCHINA
100 DATA162,1,120,173,2,220,141,0,193,169,192,141,2,220,169
110 DATA128,141,0,220,160,128,234,136,16,252,173,25,212,157
120 DATA1.193,173,26,212,157,3,193,173,0,220,9,128,141,5,193
130 DATA169,64,202,16,222,173,0,193,141,2,220,173,1,220,141
140 DATA6,193,88,96
```

CAPITOLO 5

ALTRE PRESTAZIONI

GRAFICA

Il vostro COMMODORE 64 puo' fare in grafica delle cose veramente eccezionali, infatti grazie al circuito di gestione video (detto VIC II), puo':

- passare ad alta risoluzione definendo il colore di 320 punti in orizzontale per 200 in verticale con due colori: questo modo e' detto BIT MAP STANDARD
- grafica con risoluzione 160 x 200 punti con quattro colori: questo modo e' detto BIT MAP MULTICOLORE
- definire piccoli oggetti detti SPRITE o ANIMAZIONI e movimentarli facilmente sullo schermo
- programmare caratteri diversi da quelli disponibili sulla tastiera
- eseguire lo SCROLLING rallentato del video
- spostare la memoria di schermo ad indirizzi di memoria diversi da quello standard.

Tuttavia tutte le sopraddette funzioni sono utilizzabili da chi ha gia' buona pratica di programmazione Basic e Assembler, pertanto in questo libro si approfondira' esclusivamente la parte relativa agli SPRITE.

SPRITE

Sono delle figure che vengono create e movimentate attraverso un certo numero di registri che nel VIC II vanno dalla locazione 53248 alla 53293. Ne potete realizzare fino ad otto diversi. Vedremo piu' avanti come trasformare una figura in una sequenza di dati, ora cominciamo con analizzare le operazioni da eseguire per definire ogni singolo SPRITE e vediamo tutte le possibilita' che avete a disposizione:

1) memorizzare in una zona opportuna tutti i dati da prelevare per la visualizzazione dello SPRITE
2) ogni SPRITE ha un registro detto puntatore che deve contenere l' indirizzo di memoria da cui cominciare a prelevare i dati

3) ogni SPRITE ha due registri che danno la posizione sullo schermo come coordinate X e Y (X orizzontale ed Y verticale).

   La X va da 0 a 320, mentre la Y va da 0 a 200, tuttavia per superare con la X il valore 255 sara' necessario un ulteriore registro.
4) esiste un registro in comune agli sprite che viene utilizzato per farli apparire o scomparire dallo schermo 0,
5) esistono due registri in comune agli otto sprite che provocano l' espansione delle figure in orizzontale o in verticale ottenendo cosi' un rapporto di proporzioni diverse o il raddoppio delle dimensioni
6) ogni sprite ha un registro per definire il colore
7) esiste un registro per ottenere degli sprite con 3 colori piu' quello dello sfondo ed altri due per programmare i colori
8) esiste un registro che consente di rilevare il contatto tra due sprite
9) esiste un registro per rilevare il contatto tra ogni sprite e lo sfondo
10) e' possibile decidere se lo sprite deve passare sopra o sotto cio' che e' disegnato sullo schermo

Tutte queste caratteristiche vengono di seguito riassunte nello schema sottoriportato con l' avvertenza che il numero di registro indicato deve venire sommato a 53248 (che e' la locazione di memoria di inizio del VIC II) per ottenere la locazione di memoria esatta:

N REG.	LOC.MEM	EFFETTO
0	53248	X coordinata sprite 0
1	53249	Y coordinata sprite 0
2	53250	X coordinata sprite 1
3	53251	Y coordinata sprite 1
4	53252	X coordinata sprite 2
5	53253	Y coordinata sprite 2
6	53254	X coordinata sprite 3
7	53255	Y coordinata sprite 3
8	53256	X coordinata sprite 4
9	53257	Y coordinata sprite 4
10	53258	X coordinata sprite 5
11	53259	Y coordinata sprite 5
12	53260	X coordinata sprite 6
13	53261	Y coordinata sprite 6
14	53262	X coordinata sprite 7

15	53263	Y coordinata sprite 7
16	53264	comune a tutti per X > 255
17	53265	controllo del VIC II (usare con cura)
18	53266	controllo video (usare con cautela)
19	53267	coordinata della penna ottica
20	53268	coordinata della penna ottica
21	53269	comune a tutti abilita o disabilita gli sprite (compaiono o scompaiono)
22	53270	controllo del VIC II (usare con cura)
23	53271	comune a tutti espande in verticale Y
24	53272	controlla la memoria del VIC II
25	53273	controllo delle interruzioni
26	53274	maschera per le interruzioni
27	53275	comune a tutti: priorita' sullo sfondo
28	53276	comune a tutti: modo multicolore
29	53277	comune a tutti espande in orizzontale X
30	53278	comune a tutti:collisione tra sprite
31	53279	comune a tutti:coll. sprite-sfondo
32	53280	colore bordo schermo (gia' visto!)
33	53281	colore dello schermo (gia' visto!)
34	53282	seleziona il secondo colore video
35	53283	seleziona il terzo colore video
36	53284	seleziona il quarto colore video
37	53285	primo registro multicolore (M1)
38	53286	secondo registro multicolore (M2)
39	53287	colore sprite 0
40	53288	colore sprite 1
41	53289	colore sprite 2
42	53290	colore sprite 3
43	53291	colore sprite 4
44	53292	colore sprite 5
45	53293	colore sprite 6
46	53294	colore sprite 7

Inoltre per consentire agli sprite di recuperare istantaneamente dalla memoria i dati che ne consentiranno la loro visualizzazione si utilizzano anche le seguenti posizioni di memoria (queste NON vanno sommate alla locazione di inizio del VIC II):

2040	puntatore dello sprite 0
2041	puntatore dello sprite 1
2042	puntatore dello sprite 2
2043	puntatore dello sprite 3
2044	puntatore dello sprite 4
2045	puntatore dello sprite 5
2046	puntatore dello sprite 6
2047	puntatore dello sprite 7

Ogni attivazione di funzioni sui registri comuni a tutti gli sprite viene ottenuta ponendo a 1 il Bit corrispondente:
perche' la cosa vi risulti facile disegnatevi il registro con le potenze di 2, come illustrato nella figura sottostante e sommate solo quei valori per i bit corrispondenti agli sprite che volete attivare: questo e' il numero che dovrete memorizzare (con l' istruzione POKE, e per favore non fatevi sentire ad usare il verbo Pokkare !!!).

```
 7   6   5   4   3   2   1   0
 -------------------------------
|   | 1 |   | 1 |   |   |   | 1 |
 -------------------------------
128  64  32  16   8   4   2   1
```

Provate ad esempio ad attivare in un registro gli sprite numero 0,4,6 cio' significa portare ad 1 i bit numero 0,4,6 ossia, secondo quanto detto eseguire la somma delle seguenti potenza di 2 (2↑0=1, 2↑4=16, 2↑6=64):

64+16+1=81   ed eseguire POKE V+(num reg), 81

Provate ore a disabilitare lo sprite N 6: dovrete eseguire la POKE col numero somma della potenze di 2 corrispondenti agli sprite superstiti: cioe' 16+1 = 17.
Ora finalmente si entria nel vivo della questione:

5.1.2 IL PRIMO SPRITE !

E' necessario predisporre una griglia (un foglio di carta a quadretti e' piu' che sufficente) di 24 colonne per 21 righe (ossia larga 24 ed alta 21) divisa in tre sottogriglie verticali di 8 colonne (lo spazio per il solito Byte) per 21 righe; in alto sara necessario numerare ogni colonna con le solite potenze di 2 partendo da DESTRA a SINISTRA (il contrario non da' assolutamente uno sprite capovolto !) come nella figura riportata nella pagina accanto.

Se avete una stampante a disposizione potrete ottenere la griglia necessaria col seguente programmino:

```
5 OPEN 1,4:CMD1
6 A$="128643216 8 4 2 '
7 PRINT" ";A$;A$;A$
10 FOR I=1 TO 21:N$=RIGHT$("  "+STR$(I),2):PRINT N$;
```

FIG.16. Schema dati di uno sprite raffigurante una barchetta a vela

```
20 FOR H=1 TO 24
30 PRINT "<SHIFT> O <SHIFT> P";:NEXT H:PRINT:PRINT"  ";
40 FOR H=1 TO 24
50 PRINT "<SHIFT> L <SHIFT> @";:NEXT H:PRINT:NEXT I
60 PRINT#1:CLOSE(1)
```

Riempite con un carattere le caselle che devono comparire nel disegno dello sprite come e' stato fatto nella figura che rappresenta una semplice barca a vela; fate ora le tre somme dei valori (segnati in alto allo schema come potenze di 2) corrispondenti alle caselle annerite, per i tre byte: ad esempio la riga 1 ha il primo byte completamente vuoto (percio' somma 0) il secondo byte ha la casella annerita in corrispondenza del bit n. 5 (percio' 32) il terzo e zero come il primo:

```
II riga:     I byte= 0
            II byte= 64+32=96
           III byte= 0

III riga:    I byte= 0
            II byte= 128+64+32+16
           III byte= 0
```

etc.

Alla fine di questa lunga serie di operazioni avete a disposizione 63 dati da introdurre in memoria con la funzione POKE.
La scelta delle locazioni di memoria libere e' abbastanza complicata, in quanto il punto di partenza deve essere un numero multiplo di 64 e naturalmente deve occupare una zona che non sia utilizzata dal sistema per l'elaborazione.
La piu' comoda e' l'area di memoria usata per il passaggio dati al registratore.

da 13x64=832 fino a 895
da 14x64=896 fino a 959
da 15x64=960 fino a 1023

poi se il programma Basic compresi i dati numerici ed alfanumerici occupa memoria al di sotto della posizione 16383 (detto Banco 0) avete a disposizione tutta la memoria dalla fine del Basic alla locazione suddetta sempre occupandola a partire da multipli di 64.

per superare la posizione 13683 passando al banco 1 (da 16384 a 32767) al banco 2 (32768 a 49151) od al banco 3 (49152 a 65535) bisognerebbe spostare temporaneamente la capacita'di lettura del VIC II, ma questa e' gia' cosa da addetti ai lavori.
Una volta stabilito il gruppo di 64 locazioni di memoria (nel nostro esempio da 832 a 895) in cui immagazzinare i dati, bisogna informare delle loro posizioni il puntatore dello sprite, cioe' mettere nel registro che indica al VIC II dove recuperare i dati per lo sprite (il nostro sprite e' il n. 0 ed il suo puntatore e' la posizione di memoria 2040) il valore corrispondente all'inizio dati diviso per 64 (ossia 832/64= 13)
Vediamo ora il procedimento in basic:

```
10  REM DA 0 A 62 SONO 63 DATI IL 64 NON SERVE
90  V= 53248
100 FOR N= 0 TO 62: READ Q: REM LEGGE I DATI
110 POKE 832+N,Q:NEXT:REM LI POSIZIONA DA 832 A 894
120 POKE 2040,13: REM POSIZIONA IL PUNTATORE A 832/64
```

Ora dovete assegnare la coordinata orizzontale (X) e quella verticale (Y) per definire il punto in cui devce apparire lo sprite sul video.
Questi due valori: partono dal lato sinistro per la X e vanno da 0 fino a 343 ; partono dal lato superiore per la Y e vanno da 0 fino a 249.
Il punto che definite con questi due valori distante X da sinistra e Y dall'alto coincide con l'angolo in alto a sinistra della griglia usata per disegnare lo sprite.
Per la precisione pero' ricordate che le due distanze non vengono prese dai bordi della parte visibile dello schermo, ma da un rettangolo piu' ampio in modo da consentire allo sprite di apparire nello schermo come se arrivasse dall'esterno.
I valori di X e Y, attraverso l'istruzione POKE, vengono memorizzati in una coppia di registri (c'e' una coppia per ogni sprite, ad esmpio quelli dello sprite 0 sono 0 per la X ed 1 per.la Y) tuttavia tenete presente che questi registri sono in grado di contenere un numero fino a 255; per superare questo inconveniente c'e' un registro (il n. 16) in ognuno degli otto bit serve per il corrispondente sprite per cui se tale bit e' nullo, la X e' minore di 255, quando pero' il bit diviene 1, lo sprite viene posizionato nella parte di schermo che va da 256 a 320 e la distanza dalla posizione 255 viene ancora letta nel registro della X, ad esempio per lo sprite 0

per X tra 0 e 255   il registro 0 contiene da 0 a 255
                     il bit 0 del registro 16 e' a 0.

per X tra 256 a 343 il registro 0 contiene da 0 a 87
                     il bit 0 del registro 16 e' a 1.

Posizioniamo lo sprite nel punto di coordinate X=30 Y=100 eseguendo le 2 POKE:

```
125 POKE V+0,30:POKE V+1,100
```

Ora finalmente diamo il consenso alla visualizzazione dello sprite:

```
130 POKE V+21,1
```

per fare questo si usa il registro 21 dove ognuno degli 8 bit corrisponde al relativo sprite.
Perche' lo sprite appaia il suo bit deve valere 1, perche' scompaia deve valere 0.
Il numero da mettere nella memoria del registro 21 deve essere in grado (trasformato in binario)di portare ad 1 i bit corrispondenti a tutti gli sprite che si vogliono visualizzare.
Ora, prima di passare ad altre possibilita' esaminate il programma base.

Provate in modo diretto a modificare la riga 125 (es:POKE V+0,50:POKE V+1,150)
e vedrete lo sprite spostarsi istantaneamente in un'altra parte del video.

```
90 V=53248:REM                    INIZIO VIC II
98 REM
99 REM                            LEGGE I DATI
100 FOR N = 0TO62: READ Q
108 REM
109 REM                           LI MEMORIZZA AL BLOCCO 13
110 POKE 832+N,Q :NEXT N:REM 832 = 64 * 13
118 REM          INFORMA IL PUNTATORE DELLO SPRITE 0 DELLA
119 REM                   LOCAZIONE IN CUI LEGGERE I DATI
120 POKE 2040,13
123 REM
124 REM                           INSERISCE LE COORDINATE X,Y
125 POKE V+0,30: POKE V+1,100
128 REM
129 REM                           ACCENDE LO SPRITE 0
130 POKE V+21,1
```

```
140 REM DATI DELLO SPRITE
10000 DATA   0, 32,  0,  0, 96,  0
10005 DATA   0,240,  0,  1,248,  0
10010 DATA   3,252,  0,  3,254,  0
10015 DATA   7,255,  0
10020 DATA  15,255,128, 15,255,192
10025 DATA  31,255,224
10030 DATA  31,255,240, 63,255,248
10035 DATA 127,255,255
10040 DATA 127,255,  0,255,255,248
10045 DATA 127,255,252
10050 DATA  63,255,254, 31,255,255
10055 DATA   0,  0,  0,  0,  0,  0
10060 DATA -  0,  0,  0
```

Il colore del vostro sprite probabilmente e' il bianco, questo perche' non ci siamo ancora preoccupati di assegnargli un colore posizionando il numero corrispondente al colore (come nella tabella sotto elencata) nel registro 39 (53287) che e' appositamente destinato a questa funzione per lo sprite 0

CODICE	COLORE	CODICE	COLORE
0	NERO	8	ARANCIONE
1	BIANCO	9	BRUNO
2	ROSSO	10	ROSA
3	CYAN	11	GRIGIO SCURO
4	PORPORA	12	GRIGIO MEDIO
5	VERDE	13	VERDE CHIARO
6	BLU	14	AZZURRO
7	GIALLO	15	GRIGIO CHIARO

Ora se volete modificare il colore del vostro sprite bastera' inserire nel programma la seguente istruzione:

```
124 REM COLORE GIALLO
127 POKE V+39,7
```

Se volete modificare il colore di volta in volta:

```
126 REM CAMBIO COLORE
127 FOR K=0 TO 15:POKE V+39,K:FOR H=1TO200:NEXTH:NEXTK
```

Lo stesso blocco di dati che avete inserito nel COMMODORE 64, puo' essere utilizzato anche per far comparire altri sprite oltre al primo, sara' sufficiente

inserire tutte le istruzioni necessarie per attivarli; ripetiamole:

- inserire l'indirizzo dati nel puntatore
- assegnare le coordinate ai registri corrispondenti
- assegnare il colore al registro relativo
- modificare il registro 21 per farli comparire: inserendo in tale registro un numero corrispondente:

ad esempio gli sprite 0,1,2 daranno il numero 1+2+4=7; gli sprite 0,3,6 daranno il numero:

1 + 2*2*2 + 2*2*2*2*2*2 = 73

Nel programma seguente in riga 120, con un ciclo FOR NEXT, vengono preparati i puntatori; poi nelle rige da 1125 1132 vengono assegnate le coordinate di tutti gli sprite (si poteva fare anche con un ciclo FOR NEXT ma risultava piu' difficile da capire, provate a farlo voi per esercizio).
Infine il colore viene fatto variare con la funzione RND (0) su tutti gli sprite grazie ad un ciclo FOR NEXT che ciclicamente passa in rassegna tutti gli sprite.

```
90 V=53248:REM   LOCAZIONE INIZIO VIC II
98 REM
99 REM           LEGGE I DATI
100 FORN=0TO62:READQ
108 REM
109 REM          LI MEMORIZZA AL BLOCCO 13
110 POKE832+N,Q:NEXT
118 REM
119 REM          PREPARA PUNTATORI SPRITE
120 FORI=0TO7:POKE2040+I,13:NEXT
123 REM
124 REM          INSERISCE LE COORDINATE X,Y
125 GOSUB1125:REM CON UN SOTTOPROGRAMMA
128 REM
129 REM          ACCENDE GLI SPRITE 0 FINO A 7
130 POKEV+21,255
140 :
150 FORH=0TO7:REM CAMBIA IL COLORE AD OGNI SPRITE
160 POKEV+39+H,RND(0)*16:NEXTH
170 GOTO140
1125 POKEV+0,30:POKEV+1,100
1126 POKEV+2,50:POKEV+3,110
1127 POKEV+4,70:POKEV+5,120
1128 POKEV+6,90:POKEV+7,130
1129 POKEV+8,110:POKEV+9,140
1130 POKEV+10,130:POKEV+11,150
```

```
1131 POKEV+12,150:POKEV+13,160
1132 POKEV+14,170:POKEV+15,170:RETURN
10000 DATA 0,32,0,0,96,0
10005 DATA 0,240,0,1,248,0
10010 DATA 3,252,0,3,254,0
10015 DATA 7,255,0      •
10020 DATA 15,255,128,15,255
10025 DATA 192,31,255,224
10030 DATA 31,255,240,63,255
10035 DATA 248,127,255,255
10040 DATA 127,255,0,255,255
10045 DATA 248,127,255,252
10050 DATA 63,255,254,31,255
10055 DATA 255,0,0,0
10060 DATA 0,0,0,0,0,0
```

Provate ora modificando il programma precedente ad animare gli sprite lasciando inalterato il loro colore e modificando la coordinata orizzontale:

- inserite la riga 80 per modificare il colore di fondo dello schermo (o rischiate di non vedere la barchetta azzurra).
- eliminate le righe 150 o 160 (nel listato sono state precedute da un REM e quindi non operano piu').

- modificate la riga 170 ed inserite le altre linee 180,181,182,183,190 che realizzano il cambiamento della coordinata X per ogni sprite grazie alla funzione RND programmata in modo che se il valore generato e' minore di 0.5 la barca arretra, se e' maggiore di 0.5 la barca avanza di 5 punti sul video.
- e' anche necessario, per evitare segnalazioni di errore, controllare con le righe 181,182 che il valore ottenuto di X sia contenuto nell'intervallo 0,255. Eventualmente modificate le coordinate di partenza degli sprite per evitare che si sovrappongano, come e' stato fatto nel programma.

```
80 POKE 53281,11
90 V=53248:REM                    INIZIO VIC II
98 REM
99 REM                             LEGGE I DATI
100 FORN=0TO62:READQ
108 REM
109 REM                           LI MEMORIZZA AL BLOCCO 13
110 POKE832+N,Q:NEXT
118 REM
119 REM                           PUNTATORI SPRITE
```

```
120 FORI=0TO7:POKE2040+I,13:NEXT
123 REM
124 REM                    INSERISCE LE COORDINATE X,Y
125 GOSUB1125
128 REM
129 REM                    ACCENDE GLI SPRITE 0 FINO A 7
130 POKEV+21,255
140 :
150 REMFORH=0TO7
160 REMPOKEV+39+H,RND(0)*16:NEXTH
170 FORH=0TO7
180 A=(.5-RND(0))*5:X=PEEK(V+(H*2))+A
181 IFX<=0THENX=0
182 IFX>=255THENX=255
183 POKEV+(H*2),X
190 NEXT:GOTO140
1125 POKEV+0,30:POKEV+1,100
1126 POKEV+2,50:POKEV+3,110
1127 POKEV+4,70:POKEV+5,120
1128 POKEV+6,90:POKEV+7,130
1129 POKEV+8,110:POKEV+9,140
1130 POKEV+10,130:POKEV+11,150
1131 POKEV+12,150:POKEV+13,160
1132 POKEV+14,170:POKEV+15,170:RETURN
10000 DATA 0,32,0,0,96,0
10005 DATA 0,240,0,1,248,0
10010 DATA 3,252,0,3,254,0
10015 DATA 7,255,0
10020 DATA 15,255,128,15,255
10025 DATA 192,31,255,224
10030 DATA 31,255,240,63,255
10035 DATA 248,127,255,255
10040 DATA 127,255,0,255,255
10045 DATA 248,127,255,252
10050 DATA 63,255,254,31,255
10055 DATA 255,0,0,0
10060 DATA 0,0,0,0,0,0
```

SPRITE NELLA ZONA SCHERMO DA 255 A 320

Riprendete il (primo programma sprite) programma a pag. 142 ed esaminate in particolare la riga 125, si vuole ora estendere il movimento in orizzontale dello sprite dalla colonna 320 alla colonna 0.
Oltre ad operare per la X sul registro 0 (si sta sempre parlando dello sprite numero 0 che ha come registri coordinate V+0 e V+1), bisogna mettere in gioco il registro 16 anzi piu' precisamente il bit 0 (per lo sprite 0) del registro 16, infatti quando siete giunti al valore 255 col vostro registro 0 un successivo

incremento provoca un errore? ILLEGAL QUANTITY ERROR percio' invece di incrementare il registro 0, si porta ad 1 il bit 0 del registro 16 e si ricomincia con 0 sul registro 0 dello sprite 0 (ovviamente non si va piu' da 0 a 255 ma da 0 a 343-256 = 87).
E' bene anche precisare che lo sprite con la X variabile tra 1 e 343 sparisce all'esterno del campo video sia a destra che a sinistra.
La nuova riga 125 sara' sostituita dalle seguenti:

```
124 POKE V+1,100  : REM INSERISCE LE COORDINATE X , Y
125 POKE V+16,1:POKEV,87:POKEV+21,1:FORX=87TO0 STEP(-1)
126 FORN = 1 TO 100 :NEXT :POKE V,X :NEXT
127 POKE V+16,0 :FOR X=255 TO 1 STEP(-1)
128 POKE V,X:FORN=1TO100:NEXT:NEXT:POKE V+21,0:GOTO 125
```

E' bene notare che prima di attivare il registro 21 si preparano sia il registro 16 che il registro 0 con i loro valori, se cio' non fosse effettuato, al ritorno in riga 125 dal GOTO di riga 128, si avrebbe un cattivo funzionamento dovuto al fatto che se il registro 16 viene posto a 1 (ed il registro 0 e' ancora a zero), si ha un salto dell'immagine sullo schermo.

INGRANDIMENTO DELLO SPRITE

Gli sprite possono raddoppiare le loro dimensioni, basta che nel registro 23 per la direzione X, e nel registro 29 per la Y il bit corrispondente, venga portato a 1 (provate nel programma precedente a modificare la linea 128 togliendo il GO TO 125 e aggiungete il seguente:

```
129 POKE V+23,1:GOSUB132: POKE V+29,1:GOSUB132
130 POKE V+23,0:GOSUB132: POKE V+29,0:GOSUB132:END
132 FOR I= 1 TO 200:NEXT:RETURN
```

SPRITE MULTICOLORI

Se siete disposti a perdere un po' della definizione grafica del vostro sprite potrete colorarlo con 4 colori diversi, questo metodo vi porta a dividere ogni bit in 4 caselle di 2 bit (ogni bit visualizzato sul televisore diviene un punto luminoso detto anche PIXEL) percio' ognuna di queste coppie di caselle potra' colorarsi con

uno dei 4 colori; e la scelta del colore potra' essere fatta tra:

- il colore dello sprite
- il colore di fondo
- il colore del registro 37 detto multicolore 1 (M1)
- il colore del registro 38 detto multicolore 2 (M2)

Per ottenere in un certo punto dello sprite un colore piuttosto che un altro si opera sulle coppie di pixel (bit sul video) riempiendole o meno con i caratteri a seconda del colore desiderato:

1) Si ha il colore di sottofondo se entrambe le caselle della coppia di bit sono vuote (0 , 0).
2) Si ha il colore del registro 37 (multicolore 1) se e' vuota la casella a sinistra, piena quella a destra.
(0 , 1).
3) Si ha il colore dello sprite se e' vuota la casella a destra e piena quella a sinistra (1 , 0).
4) Si ha il colore del registro 38 (multicolore 2) se entrambe le caselle sono piene. (1 , 1)

In sintesi:

(0,0 =CF) COLORE DI FONDO (0,1 =M1) MULTICOLORE 1

(1,0 =CA) COLORE ANIMAZIONE (1,1 =M2) MULTICOLORE 2

Prendete ad esempio la quarta riga dello sprite con i colori che verranno visualizzati:

00	00	00	01	11	11	10	00	00	00	00	00
CF	CF	CF	M1	M2	M2	CA	CF	CF	CF	CF	CF

E' evidente che, per ottenere un risultato di colorazione efficace bisogna ristudiare la composizione dello sprite.Si procede nel seguito a modificare l' esempio gia' visto in modo da ottenere:

- lo scafo colorato col colore di animazione (CA) (e percio' le coppie saranno tutte (10))

- la vela della barca bianca (M1) coppie (01)

- l'albero della barca nero (M2) coppie (11)

- inoltre si vogliono staccare le due vele nel punto in cui si sovrappongono usando lo stesso colore dello scafo: (sempre CA con le coppie a 10)

```
1 REM MULTICOLORE
90 V=53248:REM  INIZIO VIC II
98 REM
99 REM            LEGGE I DATI
100 FORN=0TO62:READQ
108 POKEV+37,1:POKEV+38,0:POKEV+39,8:POKEV+28,1
109 REM           LI MEMORIZZA AL BLOCCO 13
110 POKE832+N,Q:NEXT
119 REM           PUNTATORE SPRITE ED ESPANSIONE
120 POKE2040,13:POKEV+23,1:POKEV+29,1
123 REM
124 POKEV+1,100:REM INSERISCE LE COORDINATE X,Y
125 POKEV+16,1:POKEV,87:POKEV+21,1:FORX=87TO0STEP(-1)
126 FORN=1TO100:NEXT:POKEV,X:NEXT
127 POKEV+21,0:POKEV+16,0:FORX=255TO1 STEP(-1)
128 POKEV+21,1:POKEV,X:FORN=1TO100:NEXT:NEXT:POKEV+21,0:
129 GOTO 125
10000 DATA 0,48,0,0,112,0
10005 DATA 0,116,0,1,116,0
10010 DATA 1,117,0,1,117,0
10015 DATA 5,117,0
10020 DATA 5,117,64,5,117,64
10025 DATA 21,101,80
10030 DATA 21,101,80,21,89,84
10035 DATA 85,91,255
10040 DATA 85,85,0,85,85,168
10045 DATA 170,170,168
10050 DATA 42,170,170,10,170,170
10055 DATA 0,0,0
10060 DATA 0,0,0,0,0,0
```

Sara' necessario definire il colore nei registri:

con le POKE V + 37,1 colore bianco al multicolore 1 (M1)
POKE V + 38,0 colore nero al multicolore 2 (M0)
POKE V + 39,8 colore giallo dello sprite
POKE V + 28,1 accensione nel registro multicolore del bit 0 (che viene portato a 1) per il multicolore dello sprite 0

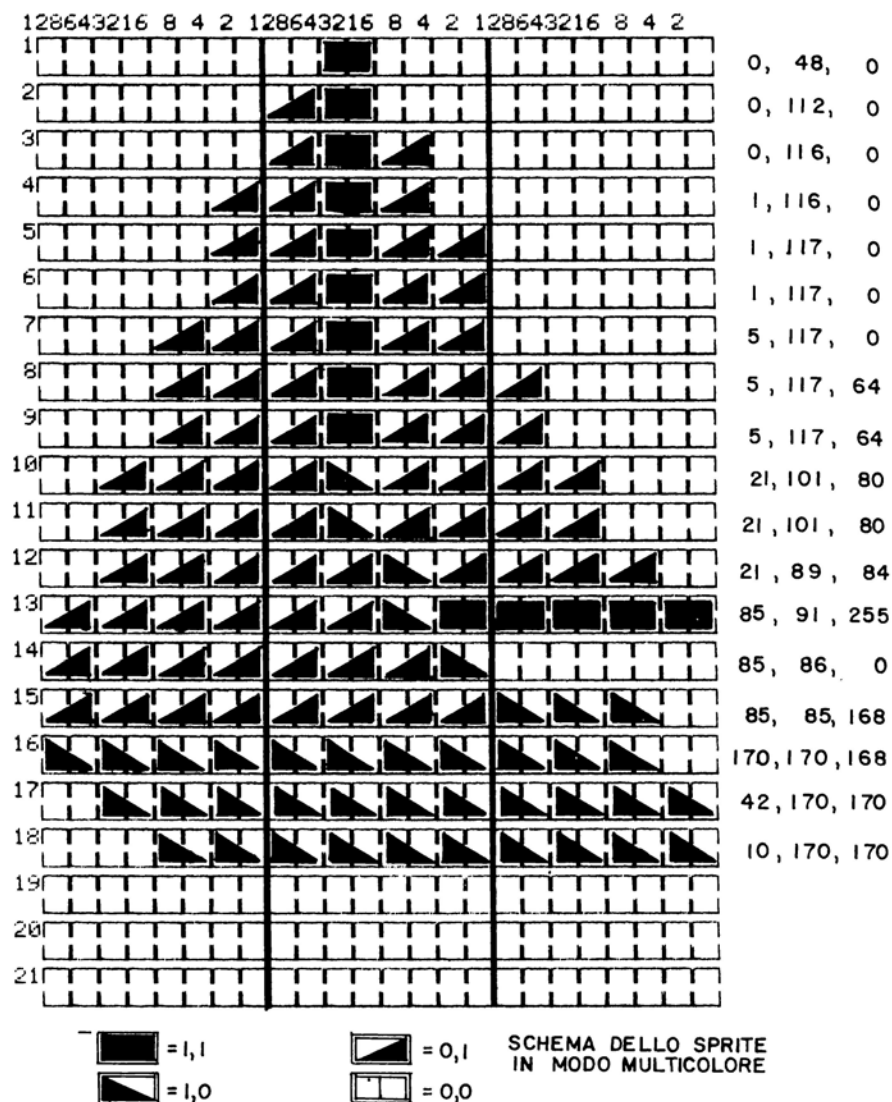



FIG.17. La Fig.16. rielaborata per generare uno sprite multicolore

COME ANIMARE LO SPRITE

Una caratteristica molto interessante del VIC II e' proprio dovuta al fatto che un certo sprite venga visualizzato come previsto in un certo blocco di dati che vengono reperiti nella memoria attraverso un registro puntatore (2040 per lo sprite 0 - 2041 per lo sprite 1 etc.).
Se in quel registro, che nei programmi visti in precedenza conteneva 13 per indicare la posizione di memoria 13*64 = 832, si dovesse memorizzare un altro numero (es.14) lo sprite cambierebbe istantaneamente di forma, raffigurando, secondo il meccanismo di calcolo gia' esposto, il nuovo blocco di dati situati da 13*14 = 896 in avanti.
In pratica un semplice cambiamento nel contenuto del registro puntatore puo' modificare il disegno di tutto lo sprite, trasformandosi in un altro disegno (se nel blocco 14 e' stata memorizzata una serie di dati) o in qualcosa di incomprensibile se nel corrispondente blocco non e' stato memorizzato niente.

Usiamo questo metodo per esaminare ad esempio alcune parti della memoria:

```
90 V=53248
91 POKEV+28,1
92 POKEV+37,1
93 POKEV+38,0
94 POKEV+39,8
100 POKEV,150
110 POKEV+1,150
120 POKEV+21,1
130 POKEV+23,1
140 POKEV+29,1
150 PRINT"<SHIFT><CLR/HOME>"
160 INPUT I:POKE 2040,I:GOTO 160
```

Ora rispondete 1,2,3 etc.. Noterete che sia per 2 , per 3 che per 7 esistono degli sprite in movimento. Cio' e' dovuto al fatto che il calcolatore sta elaborando e quindi ha delle posizioni di memoria, che state visualizzando con lo sprite, in cui vengono variati i valori dei bit.

Nel blocco 9 non ci sono dati, percio' lo sprite scompare, cosi' pure in 11, 13, 14 e 15.
Ora date 16 e portate il cursore in alto <HOME>: state visualizzando con lo sprite un'elaborazione dei dati dello schermo (16*64 = 1024 e lo schermo va proprio da 1024 a 2023); provate a muovere il cursore nella prima

riga (e fino al 23~ carattere della seconda riga) o a battere dei tasti e provocherete un cambiamento nello sprite (se usate i caratteri grafici lo sprite si riemp'e di piu':perche'?).
Finito l'esperimento riprendete il programma della barca in movimento, ed aggiungete un altro blocco di dati (il 14~) uguale al precedente salvo che la barca e' stata abbassata di due righe (percio' 6 zeri iniziali per le prime due righe, seguiti da tutti i dati precedenti e da sei zeri di meno in fondo).
Questi dati sono stati aggiunti al programma da riga 19999 a riga 20060. Il programma abilita sempre e solo lo sprite 0 modificando in riga 125 il blocco di dati puntati dal puntatore (2040) secondo una funzione RND(0).
Naturalmente e' stato aggiunto un secondo ciclo di lettura e memorizzazione dati in riga 102, 103 per caricare il secondo blocco di dati.
L'immagine ottenuta dal programma non e' ancora un'animazione vera e propria, ma rende sufficientemente l'idea di come operare. Per esercizio provate di seguito a disegnare la stessa barca inclinandola leggermente, memorizzate i nuovi dati nel blocco 15 o nei blocchi piu' elevati (l'importante e' non coprire il basic e lo schermo) e realizzate un ciclo in cui il registro 2040 punti ciclicamente tutti i blocchi.

```
1 REM SPRITINIZ
90 V=53248:REM   INIZIO VIC II
91 POKEV+32,5 :POKEV+33,5 :POKEV+28,0
92 POKEV+37,1 :POKEV+38,8 :POKEV+39,9
100 FORN=0TO62:READQ
110 POKE832+N,Q:NEXT N
111 FORN=0TO62:READQ
112 POKE896+N,Q:NEXT N
119 REM           PUNTATORE SPRITE
120 POKE2040,13
123 POKEV+1,229:REM INSERISCE LE COORDINATE X,Y
124 POKEV+16,1:POKEV,87:POKEV+21,1:FORX=87TO0STEP(-1)
125 IFRND(0)<.5THENPOKE2040,RND(0)+13.5
126 FORN=1TO100:NEXTN:POKEV,X:NEXTX
127 POKEV+16,0:FORX=255TO1 STEP(-1)
128 IFRND(0)<.5THENPOKE2040,RND(0)+13.5
129 POKEV,X:FORN=1TO100:NEXTN:NEXTX:POKEV+21,0:GOTO124
1000 DATA0,6,0,0,6,96
1010 DATA0,12,96,0,211,128
1020 DATA3,60,0,4,144,0
1030 DATA8,16,0,17,160,0
1040 DATA38,96,0,72,7,128
```

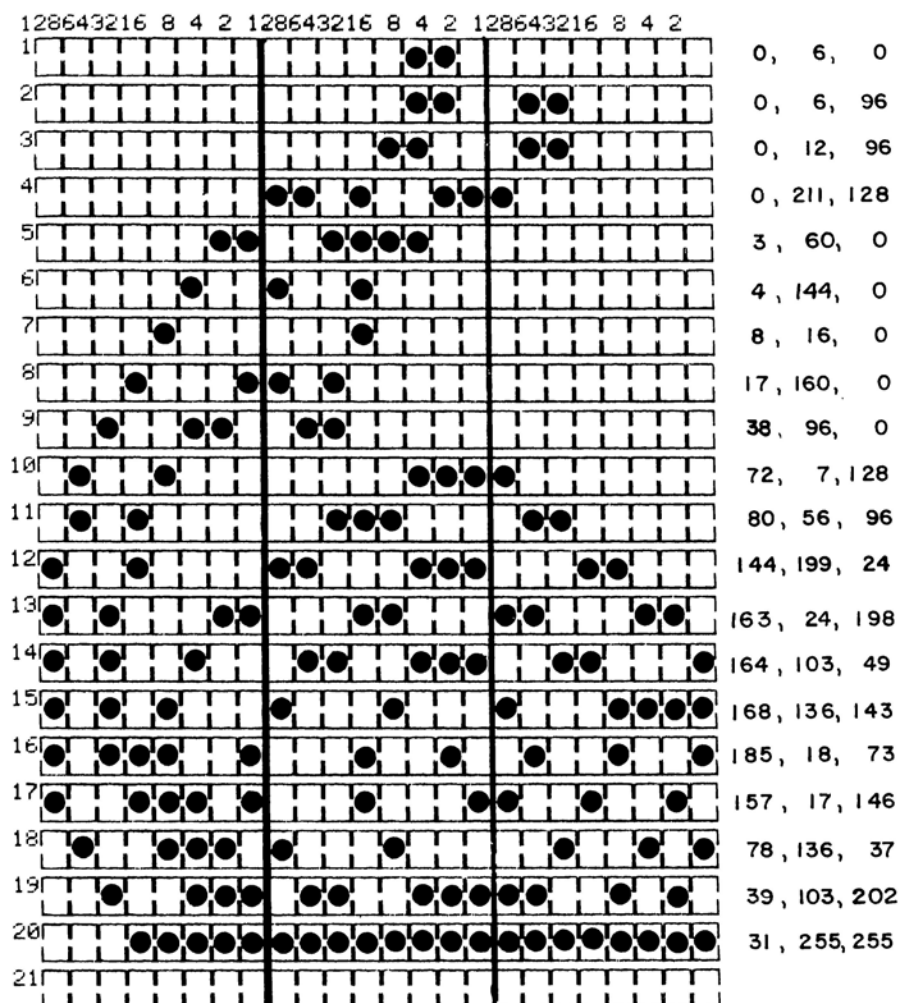



FIG.18. Prima serie di dati per animare una lumaca sprite !

```
1050 DATA80,56,96,144,199,24
1060 DATA163,24,198,164,103,49
1070 DATA168,136,143,185,18,73
1080 DATA157,17,146,78,136,37
1090 DATA39,103,202,31,255,255
1100 DATA0,0,0
1110 DATA0,24,0,0,25,128
1120 DATA0,33,128,3,206,0
1130 DATA12,112,0,17,64,0
1140 DATA36,64,0,74,128,0
1150 DATA83,0,0,144,7,128
1160 DATA160,56,96,160,199,24
1170 DATA163,24,198,164,103,49
1180 DATA168,136,143,185,18,73
1190 DATA157,17,146,78,136,37
1200 DATA39,103,202,31,255,255
1210 DATA0,0,0
```

PRIORITA' E COLLISIONE TRA SPRITE.

Avete avuto modo di notare che lo sprite si muove sullo schermo indipendentemente dai caratteri su di esso contenuti; potete pero' scegliere se farlo muovere in modo da coprire con la sua figura i caratteri (e questo sara' l'effetto gia' da voi osservato) oppure se farlo muovere nascosto dietro i caratteri, cio' vi consentira' di ottenere degli effetti di profondita' nella programmazione della grafica. Il meccanismo e' quello di sempre: si utilizza il registro 27 dove ogni bit ha una corrispondenza con lo sprite dello stesso ordine;
- se il bit e' 0 lo sprite passa sopra il carattere
- se il bit e' 1 lo sprite e' nascosto dai caratteri.

Tenete presente che se lo sprite ha un foro nel suo disegno ed ha priorita' maggiore dello schermo (cioe' copre i caratteri) avrete modo di vedere il contenuto dello schermo attraverso il foro, come se fosse una finestra.
Gli sprite sono dotati di una priorita' anche tra loro e questa non e' modificabile: uno sprite con numero basso copre sempre uno sprite con numero piu' alto.
Infine e' possibile rilevare se uno sprite e' entrato in collisione con lo sfondo o con un altro sprite: la collisione, in particolare, non viene rilevata quando si ha contatto tra il rettangolo che inviluppa gli sprite ma effettivamente quando due parti non zero di sprite o di sprite e caratteri di schermo si toccano, con la sola eccezione del modo multicolore per il colore del registro 37 (M1), percio' se prevedete di rilevare

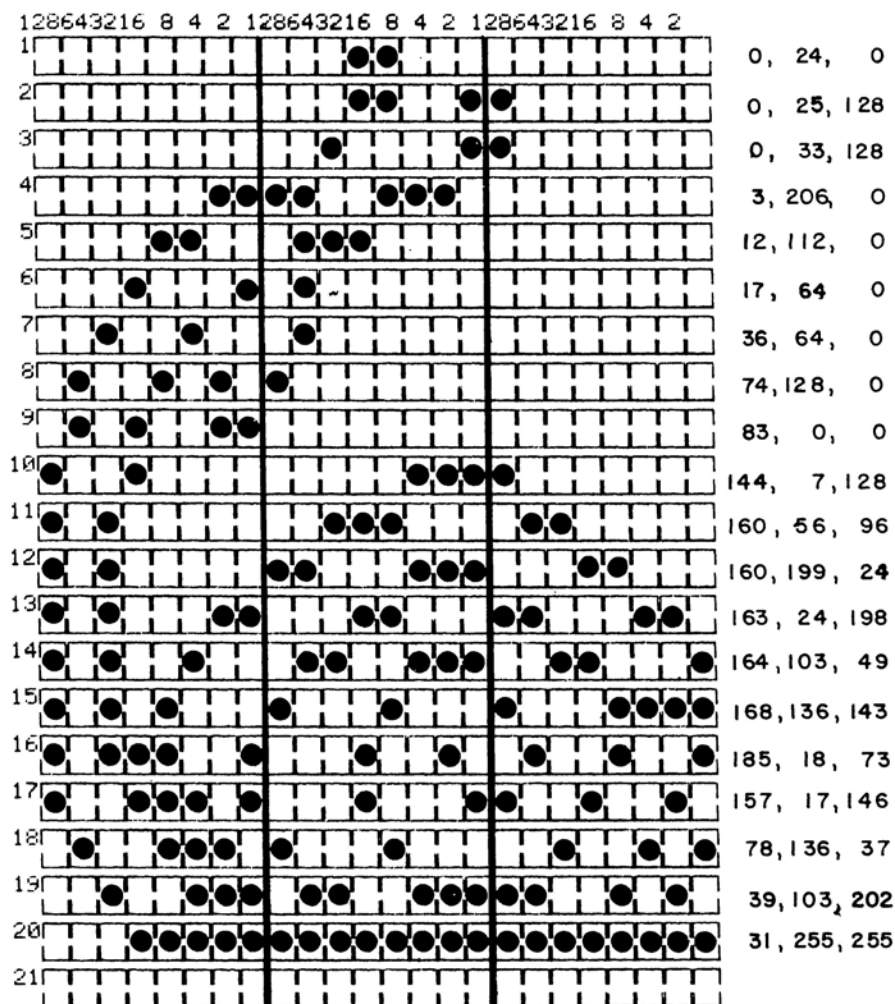



FIG.19. Come la Fig.18. ma differente nella posizione della testa

contatti tra sprite o tra sprite e schermo tenete il multicolore 1 (quello della coppia di bit 0,1) all'interno dello sprite.
In caso di collisione tra sprite e schermo il bit corrispondente allo sprite nel registro 31 diviene 1 ed e' leggibile con la funzione PEEK (numero di registro), successivamente alla lettura tutti i bit vengono azzerati.

In caso di collisione tra due sprite i bit corrispondenti e gli sprite nel registro 30 vengono portati ad 1, sono leggibili con la funzione PEEK e successivamente alla lettura vengono subito azzerati. Questo metodo presenta dei pericoli per il programmatore:

- non potete leggere due collisioni diverse con due PEEK successivi, perche' la seconda troverebbe il registro completamente azzerato
- e' necessario definire una variabile con le lettura del registro effettuata dalla PEEK
- e' necessario inoltre usare le funzioni logiche per riuscire a capire velocemente quale sprite e' entrato in collisione operando sulla variabile definita al punto precedente.

Si supponga che lo sprite 3 sia entrato in collisione con lo sprite 5, che lo sprite 6 sia entrato in collisione con lo sfondo, e che in precedenza non si siano avute collisioni:

il registro 30 conterra': (bin) 0010100 = 40 (dec)
il registro 31 conterra': (bin) 0100000 = 64 (dec)

Se ora eseguite un AND con le seguenti maschere

```
128 (dec) = 10000000 (bin) (7~ sprite)
 64 (dec) = 01000000 (bin) (6~ sprite)
 32 (dec) = 00100000 (bin) (5~ sprite)
 16 (dec) = 00010000 (bin) (4~ sprite)
  8 (dec) = 00001000 (bin) (3~ sprite)
  4 (dec) = 00000100 (bin) (2~ sprite)
  2 (dec) = 00000010 (bin) (1~ sprite)
  1 (dec) = 00000001 (bin) (0  sprite)
```

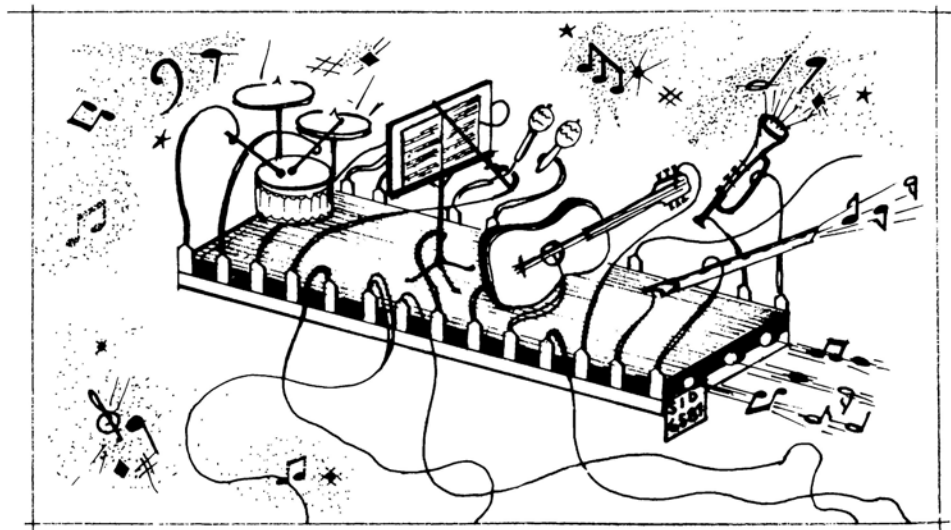
ottenete un risultato diverso da zero in caso di collisione, nullo se non e' avvenuta collisione.

ANCORA SUL DISEGNO DEGLI SPRITE

Per poter ottenere i dati degli sprite direttamente dal calcolatore potrete battere il programma segunte, dove tutto dovrebbe ormai risultare chiaro, ad eccezione di due locazioni di memoria (211 e 214 che vengono lette con l' istruzione PEEK e danno la posizione del cursore in quell' istante sullo schermo:
- in locazione 211 e' contenuto il numero della colonna
- in locazione 214 quello della riga.

```
90 V=53280-32:POKE650,128
100 PRINT"□":POKEV+32,0:POKEV+33,0
110 POKEV+16,1:POKEV,20:POKEV+1,195
120 POKEV+21,1:POKEV+23,1:POKEV+29,1
130 POKEV+39,8:POKE2040,13
200 FORI=55296TO56295:POKEI,13:NEXTI
210 FORH=0TO20
220 FOR I=1024+40*H TO1023+24+40*H
230 POKEI,46:NEXTI:NEXTH
240 DIMA(21,24),B(21,3)
300 A1$="▌▌▌▌▌▌▌▌▌▌▌▌▌▌▌▌▌▌▌▌▌▌▌▌▌▌"
301 A$="▩"+A1$
310 PRINTA$;"▥▩1▬ DATI"
330 PRINTA$;"▩▩2▬ SPRITE":PRINTLEFT$(A$,11)+MID$(A$,1,10)+"▩▩";CHR$(115);
340 GETQ$:IFQ$=""THEN340
350 IFQ$="1"THENGOSUB2000
360 IFQ$="2"THENGOSUB1000
361 IFQ$=CHR$(17) AND PEEK(214)<20THENGOSUB3000: PRINT"▌▩";CHR$(115);
362 IFQ$=CHR$(29) AND PEEK(211)<24THENGOSUB3000: PRINTCHR$(115);
363 IFQ$=CHR$(145) THENGOSUB3000: PRINT"▌□";CHR$(115);
364 IFQ$=CHR$(157) AND PEEK(211)>1 THENGOSUB3000: PRINT"▌▌";CHR$(115);
367 IFQ$="/"THENQ1$="/":GOSUB3000
370 Q$="":GOTO340
1000 FORH=0TO20
1010 FORK=0TO2
1020 FORI=0TO7:S=1024+I+K*8+40*H
1030 S2=PEEK(S):
1040 IFS2=32 OR S2=46 THENA(H,I+K*8)=0:GOTO1051
1050 A(H,I+K*8)=2↑(7-I)
1051 NEXTI:NEXTK:NEXTH
1100 FORH=0TO20
1110 FORK=0TO2:N=0
1120 FORI=0TO7
1130 N=N+A(H,I+K*8):NEXTI:B(H,K)=N
1140 NEXTK:NEXTH
1200 M=0:FORH=0TO20
1210 FORK=0TO2
1220 POKE832+M,B(H,K):M=M+1:NEXTK:NEXTH
1230 RETURN
2000 POKEV+21,0:PRINT"▩▩▩";:FORH=0TO20:G$=""
2010 FORK=0TO2
2020 H$="    "+STR$(B(H,K))
2021 G$=G$+RIGHT$(H$,4):NEXTK
2030 PRINTA1$;G$:NEXTH:Q$=""
2040 GETQ$:IFQ$=""THEN2040
2050 POKEV+21,1:PRINT"▩";CHR$(115);:GOSUB4000
2060 RETURN
3000 IFQ1$="/"THENPRINT"▌▌";CHR$(166);:Q1$="":GOTO3020
3010 PRINT"▌▌";CHR$(46);:RETURN
3020 IFPEEK(211)<24THENPRINTCHR$(115);:RETURN
3030 POKE211,0:PRINT"▩"CHR$(115);
3040 IFPEEK(214)>20THEN PRINT"▌▌ ▩";CHR$(115);:RETURN
3050 RETURN
4000 PRINT"▩▩":FORH=1TO21
4010 PRINTA1$+"                    ":NEXT
4020 RETURN
```

Come avete visto per la grafica, anche l'emissione di suoni e' affidata nel computer ad un circuito particolare chiamato "SID" che ha inizio alla locazione di memoria S=54272 ed e' dotato di 29 registri (quindi fino a 54300) di questi i primi 25 (25 vuol dire da 0 a 24 compresi) sono accessibili solo per scrittura (W=WRITE), cioe' possono essere impostati ad un dato valore o modificati nel loro valore usando l'istruzione POKE, ma NON POSSONO ESSERE LETTI con l'istruzione PEEK, gli altri quattro (i numeri 25-26-27-28) sono registri di sola lettura (cioe' possono essere letti con PEEK), ma non ha senso inserirvi dei dati), di questi i primi due sono usati dal sistema mentre gli altri due servono per effetti speciali.
Il SID e' studiato in modo da poter provocare il suono di 3 voci separate anche contemporaneamente (avete in mano un complesso di tre possibili strumenti da programmare e non e' neanche necessario che siate dei musicisti, sara' sperabile che conosciate almeno le note ...!). Ogni voce necessita di alcuni dati che vi permetteranno di assegnarle la sua personalita' e si vedranno piu' avanti in dettaglio; questi dati vengono comunque memorizzati nei registri del SID che sono suddivisi in quattro parti:

- da 0 a 6 per la prima voce
- da 7 a 13 per la seconda voce
- da 14 a 20 per la terza voce
- da 21 a 24 controllano le tre voci insieme.

Per semplificare la trattazione dei registri si indica con S=54272 la locazione di inizio memoria del SID per cui si intende che la loro numerazione va sommata al numero S.

54272		0	voce 1	FREQUENZA NOTA BYTE BASSO
54273		1	voce 1	FREQUENZA NOTA BYTE ALTO
54274		2	voce 1	AMPIEZZA DI PULSAZIONE BYTE BASSO
54275		3	voce 1	AMPIEZZA DI PULSAZIONE BYTE ALTO
54276		4	voce 1	FORMA D'ONDA (17, 33, 65, 129)
54277		5	voce 1	ATTACK - DECAY
54278		6	voce 1	SUSTAIN - RELEASE
54279	7+0	7	voce 2	FREQUENZA NOTA BYTE BASSO
54280	7+1	8	voce 2	FREQUENZA NOTA BYTE ALTO
54281	7+2	9	voce 2	AMPIEZZA DI PULSAZIONE BYTE BASSO
54282	7+3	10	voce 2	AMPIEZZA DI PULSAZIONE BYTE ALTO
54283	7+4	11	voce 2	FORMA D'ONDA (17, 33, 65, 129)
54284	7+5	12	voce 2	ATTACK - DECAY
54285	7+6	13	voce 2	SUSTAIN - RELEASE
54286	14+0	14	voce 3	FREQUENZA NOTA BYTE BASSO
54287	14+1	15	voce 3	FREQUENZA NOTA BYTE ALTO
54288	14+2	16	voce 3	AMPIEZZA DI PULSAZIONE BYTE BASSO
54289	14+3	17	voce 3	AMPIEZZA DI PULSAZIONE BYTE ALTO
54290	14+4	18	voce 3	FORMA D'ONDA (17, 33, 65, 129)
54291	14+5	19	voce 3	ATTACK - DECAY
54292	14+6	20	voce 3	SUSTAIN - RELEASE
54293		21	LIMITE FILTRO DI FREQUENZA BYTE BASSO	
54294		22	LIMITE FILTRO DI FREQUENZA BYTE ALTO	
54295		23	SCELTA FILTRO E RISONANZA	
54296		24	VOLUME E ACCENSIONE FILTRI	
54297		25	LETTURA DATI X PADDLE	
54298		26	LETTURA DATI Y PADDLE	
54299		27	LETTURA 8 BIT ALTI OSCILLATORE VOCE 3	
54300		28	LETTURA USCITA GEN. INVILUPPO VOCE 3.	

STRUTTURA DEL SUONO

Vi sara' capitato di premere il tasto di un pianoforte ed ascoltarne l'emissione sonora:

- il volume del suono sale rapidamente al valore massimo
- successivamente si attenua
- persiste fino all'istante in cui togliete il dito dal tasto
- poi scende a zero rapidamente.

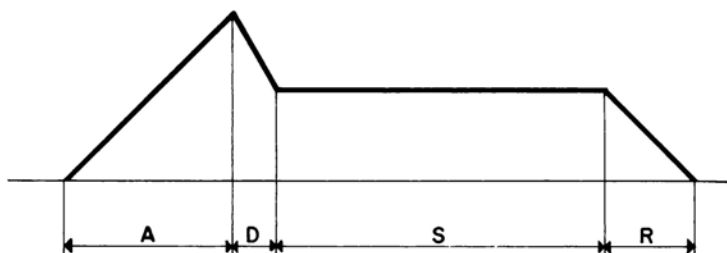
Per poter riprodurre lo stesso effetto sonoro col calcolatore dovrete definire i quattro valori delle fasi di sviluppo del suono che si susseguono come nella figura n. 20

GENERATORE D'INVILUPPO DEL SUONO

FIG.20.

A e' il tempo in cui il suono raggiunge il suo massimo volume
D e' il tempo in cui il suono si porta al volume medio di persistenza
S e' il tempo in cui il suono continua a rimanere attivo
R e' il tempo in cui il volume si azzera.

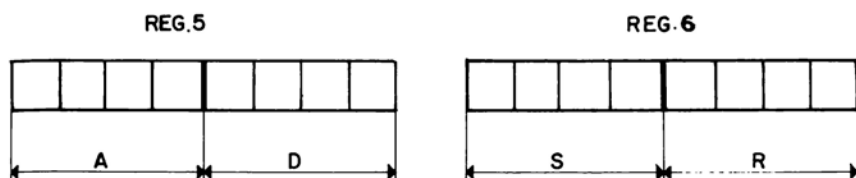
In inglese i termini corrispondenti alle quattro lettere ADSR sono i seguenti:

ATTACK DECAY SUSTAIN RELEASE

Poiche' una traduzione che renda bene il concetto di queste quattro parole mantenendo le stesse lettere iniziali non e' possibile, di seguito si indicheranno sempre le iniziali maiuscole o le stesse parole inglesi. Le quattro fasi devono, tradotte in numero e per ogni voce, essere memorizzate nei registri 5-6, 12-13, 19-20 con la tecnica che viene di seguito esposta per i registri 5 e 6, ma risulta identica anche per le altre due coppie di registri.
Il registro 5 viene diviso in due parti di 4 Bit:

- la parte alta viene usata per memorizzare la durata dell'ATTACK
- la parte bassa per memorizzare la durata del DECAY

Il registro 6 e' diviso anch'esso in due parti:

- la parte alta contiene SUSTAIN
- la parte bassa contiene RELEASE

Poiche' 4 Bit possono rappresentare valori da 0 a 15 avrete la possibilita' di scegliere un valore per ognuna delle quattro fasi interno a questo intervallo. Tuttavia per memorizzare A, una volta definito il suo valore, nella parte alta del registro 5 bisognera' ricordarsi di eseguire il prodotto per 16 che, essendo la quarta potenza di 2, vi dara' un risultato in grado di occupare effettivamente la parte alta del registro. La medesima operazione sara' effettuata con il valore da assegnare a SUSTAIN che va memorizzato nel registro 6; per i valori di D e di R (il primo nel registo 5, il secondo nel registro 6) non devono essere eseguiti calcoli, essi vanno rispettivamente sommati ai valori di A e di S (ottenuti dalla moltiplicazione sopraddetta) rispettivamente.
Le durate dei quattro valori di ADSR non sono proporzionali ai valori da 0 a 15, ma variano secondo il seguente schema:

valore	tempo ATTACK	tempo DECAY-RELEASE
0	2 ms	6 ms
1	8 ms	24 ms
2	16 ms	48 ms
3	24 ms	72 ms
4	38 ms	114 ms
5	56 ms	168 ms
6	68 ms	204 ms
7	80 ms	240 ms
8	100 ms	300 ms
9	250 ms	750 ms
10	500 ms	1.5 s
11	800 ms	2.4 s
12	1 s	3 s
13	2 s	9 s
14	5 s	15 s
15	8 s	24 s

I dati ADSR vengono anche chiamati GENERATORE DI INVILUPPO della voce ed in genere definiscono assieme alla forma dell'onda l'emissione sonora.

Il suono si propaga nell'aria provocandone lo spostamento in modo ondulatorio, cioe' un corpuscolo investito dall'emissione sonora si muovera' avanti e indietro, rispetto al punto in cui si trova, piu' o meno violentemente in funzione della potenza sonora emessa, e la velocita' con cui eseguira' un intero ciclo di movimento dipendera' dalla nota emessa; questo meccanismo viene indicato normalmente come onda sonora. Queat'ultima e' simile a quella che potete vedere se provocate una perturbazione in uno stagno lasciandovi cadere un oggetto: se l'oggetto e' piccolo le onde non saranno molto marcate, mentre se l'oggetto e' di dimensioni notevoli le onde saranno piu' evidenti.
Se si dovesse disegnare in un grafico l'entita' dello spostamento del corpuscolo attorno al punto in cui si trova rispetto al trascorrere del tempo si otterrebbe il disegno dell'onda sonora; tale onda ha in genere la forma di una sinusoide e la sua caratteristica principale e' il tempo che impiega a svilupparsi completamente in modo da ripetersi identica per un secondo ciclo: questo tempo e' definito periodo dell'onda e si misura in secondi o frazioni di secondi. Se si misura l'inverso: cioe' il numero di volte che l'onda esegue il suo ciclo nell'unita' di tempo si definisce finalmente la FREQUENZA (che si misura in Hertz ossia cicli al secodo).
Il suono emesso da uno strumento e' sempre caratterizzato da una frequenza fondamentale, che e' quella corrispondente alla nota suonata, ma e' in genere accompagnato da altre onde sonore che hanno una frequenza multipla intera (doppia, tripla etc.) di quella fondamentale: queste onde si chiamano armoniche superiori e sono importanti perche' la loro presenza (in quantita') da' una caratteristica al suono (detto Timbro) che contraddistingue tra loro gli strumenti musicali. Il risultato finale dell'onda assieme alle sue armoniche e' una forma non sinusoidale ottenuta sommando le varie onde come esemplificato nella figura num. 21

Il vostro COMMODORE 64 ha la possibilita' di emettere onde secondo tre tipi di forme diverse, non armoniche, ma geometriche che vengono stabilite attivando i Bit n. 6, 5, 4 (occhio! i Bit non i Byte o i registri):

- del registro n. 4 per la prima voce
- del registro n. 11 per la seconda voce
- del registro n. 18 per la terza voce

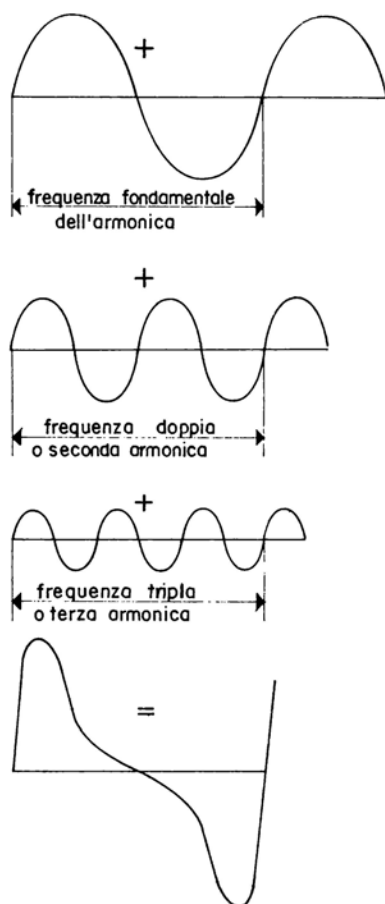



FIG.21. Esempio di onda sonora composta da frequenza fondamentale ed armoniche superiori

La prima forma d'onda e' chiamata TRINGOLARE, si attiva portando a 1 il Bit n. 4 del registro corrispondente (ossia inserire il 17 nel registro). Le armoniche possedute da tale forma d'onda sono casuali, la loro quantita' e' proporzionale al reciproco del quadrato del numero d'ordine dell'armonica (l'armonica del secondo ordine sara' presente in quantita' 1/(2*2) = 1/4,

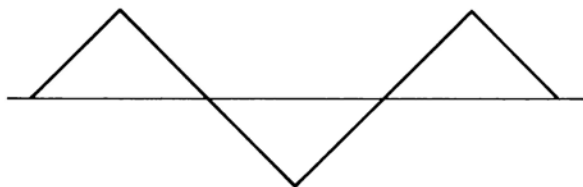
l'armonica del terzo ordine in quantita' 1/(3*3) = 1/9 e cosi' via), questa e' l'onda piu' vicina come forma ad un'onda sinusoidale percio' produrra' anche l'emissione piu' melodiosa.

forma d'onda triangolare
BIT. n°4 = I (DEC. 17)

FIG. 22.

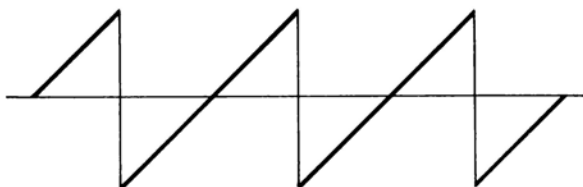
La seconda forma d'onda e' chiamata DENTE DI SEGA, e' ancora triangolare ma con un lato diritto e con un vertice piu' acuto: si attiva portando ad 1 il Bit n. 5 del registro corrispondente (si inserisce 33 nel registro). Le armoniche possetude da tale forma d'onda sono in quantita' proporzionale all'inverso dell'ordine dell'armonica (la seconda 1/2, la terza 1/3 e cosi' via). Essendo la forma molto acuta il suono sara' prevalentemente metallico come quello generato da una corda (arpa, banjo etc.).

forma d'onda a dente di sega
BIT. n° 5 = I (DEC. 33)

FIG. 23.

La terza forma d'onda e' chiamata ONDA QUADRA AD IMPULSO VARIABILE, la sua struttura e' rettangolare con la parte superiore e la parte inferiore di due lunghezze (meglio durate) diverse. Il valore di queste due lunghezze deve essere definito dal musicista, scusate dal programmatore

(anzi da chi mài riuscira' a capire le onde ad impulso variabile). La definizione viene eseguita sull'ampiezza della parte superiore con due registri per ognuna delle tre voci (2-3 per la prima, 9-10 per la seconda, 16-17 per la terza); nel registro col numero piu' basso si memorizza un numero da 0 a 255 che chiameremo L (perche' per gli inglesi L=LOW=BASSO); nel registro col numero piu' alto si puo' memorizzare un numero da 0 a 15, chiameremo questo secondo numero con H (H=HIGT=ALTO). Il COMMODORE 64 usa i due registri insieme in modo da ottenere un numero che va

da 0 fino a H * 256 + L

cioe' 15 * 256 + 255 = 4095

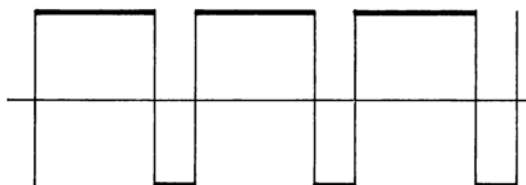
che corrisponde, se fate mente locale a quanto detto sull'algebra di Boole, al massimo numero rappresentabile con 12 Bit.

forma d'onda ad impulso variabile
BIT. n°6 = 1 (DEC. 65)

FIG. 24.

Poiche' la frequenza dell'onda quadra ad impulso variabile corrisponde proprio al valore 4095, cioe' la lunghezza della parte bassa piu' la parte alta e' sempre costante e vale 4095, se nei due registri memorizzate il numero 2048 (il registro basso conterra' 0 e l'alto conterra' 8) avrete diviso in due l'ampiezza dell'onda che risultera' quindi avere la parte positiva di lunghezza pari alla negativa, il suono in tal caso sara' abbastanza simile ad un organo, mentre per valori molto diversi tra l'ampiezza dell'onda al positivo da quella al negativo il suono sara' decisamente metallico.

I registri delle forme d'onda controllano anche l'emissione del suono nel senso che quando inserite il valore nel registro il suono comincia ad essere emesso,

quando cambiante il valore lo stesso viene arrestato (17 produce il suono ad onde triangolari, 16 lo blocca).
Provate ora i seguenti programmi per poter capire bene come ADSR e le forme d'onda riescano a modificare il suono di una stessa nota, successivamente si comincera' a far suonare il computer. Il programma e' fatto come segue:

10 si definisce l'inizio memoria del SID
11 si azzerano tutti i registri del SID (visto che non li potete leggere non potete neanche prevedere che cosa contengono)
20 per poter suonare una nota bisogna alzare il volume! ed il massimo e' 15
30-31 definizione di una nota (il LA)
50 ciclo di definizione e variazione dell'ATTACK (da 0 a 15) tenendo DECAY a 0
60 definizione dell'S-R (messo a 0)
70 accensione della forma d'onda (trangolare = 17) e ciclo di ritardo per udire l'emissione sonora
80 spegnimento della forma d'onda (16 = spento) e ciclo di ritardo per staccare una nota dalla successiva.

```
10 S=54272
11 FOR K=0 TO 24:POKE S+K,0:NEXT
20 POKE S+24,15
30 POKE S,49
31 POKE S+1,28
50 FOR I=0 TO 15:POKE S+15,I
60 POKE S+6,0
70 POKE S+4,17: FOR N=1 TO 200:NEXT
80 POKE S+4,16: FOR N=1 TO 50 :NEXT:PRINT I;:NEXT
90 STOP
```

Vedrete lanciando il programma come la variazione dell'ATTACK modifichi il suono e lo renda meno brusco man mano che il valore sale a 15.
Per constatare l'effetto della variazione di DEKAY modificate la riga 50 con la seguente:

```
50 FOR I=1 TO 15:POKE S+5,I*16-1
```

Per variare SUSTAIN modificate nel programma le due seguenti righe:

```
50 POKE S+5,15
60 FOR I= 0 TO 15:POKE S+6,I
```

Infine per ascoltare RELEASE (anche se si nota poco) modificate la riga 60:

```
60 FOR I= 1 TO 15:POKE S+6,I*16-1
```

Provate ora a variare la frequenza usando lo stesso programma con modificate le linee 30 e 31:

```
30 FOR H=0 TO 255 : POKE S+1,H
31 FOR L=0 TO 255 STEP 32 :POKE S,L
```

Per rendere il programma piu' veloce potete togliere il ciclo di ritardo in riga 80 ed accorciare quello in riga 70.
Ora che avete ascoltato le capacita' di escursione sonora del vostro computer, provate a modificare la forma d'onda a triangolare a dente di sega:

- in riga 70 sostituite 17 con 33 (accensione)
- in riga 80 sostituite 16 con 32 (spegnimento)

Poi ancora provate a scoprire cosa producono le onde quadre:

```
10 S = 54272
11 FOR K=0 TO 24:POKE S+K,0:NEXT
20 POKE S+24,15
30 POKE S,49
31 POKE S+1,28
50 POKE S+5,15
60 POKE S+6,15
65 FOR H=0 TO 15:POKE S+3,H
66 FOR K=0 TO 255 STEP 8
67 POKE S+2,K
70 POKE S+4,65:FOR N=1 TO 200:NEXT N
80 POKE S+4,64:FOR N=1TO50:NEXT N:NEXT K:PRINT H:NEXT H
```

Nel programma precedente si e' selezionata la forma d'onda in riga 70 (POKE S+4,65 spenta poi con POKE S+4,64 in riga 80) e si sono variate le proporzioni tra le ampiezze delle due parti positiva e negativa dell'onda utilizzando:

- il registro 2 ,che ha ricevuto man mano valori da 0 a 255 a passo di 8
- la parte alta del registro 3 che funziona come parte alta di tutto il numero ottenuto accodando il registro 2 al 3.

La combinazione dei 4 valori ADSR e l'uso appropriato delle forme d'onda puo' generare dei suoni molto simili agli strumenti musicali oppure dei suoni di strumenti non esistenti:

- VIOLINO : A=5 ; D=8 ; S=5 ; R=9 ossia:
  POKE S+5,88:POKE S+6,89
  con onda a dende di sega (POKE S+4,33)

- VIOLONCELLO : A=10; D=3 ; S=10; R=4 ossia:
  POKE S+5,163:POKE S+6,164
  con onda a dende di sega (POKE S+4,33)

- XILOFONO : A=0 ; D=9 ; S=0 ; R=9 ossia:
  POKE S+5,9:POKE S+6,9
  con onda triangolare (POKE S+4,17)

-ORGANO : A=0 ; D=15 ; S=15 ; R=0 ossia:
  POKE S+5,15:POKE S+6,240
  con onda quadr (POKE S+4,17)
  POKE S+2,0 POKE S+3,8

- TROMBA : A=0 ; D=15 ; S=0 ; R=15 ossia:
  POKE S=5,15:POKE S+6,15
  con onda a dende di sega (POKE S+4,33)

- FLAUTO : A=2 ; D=1 ; S=3 ; S=0 ; R=15 ossia:
  POKE S=5,45:POKE S+6,15 (note acute!)
  con onda triangolare (POKE S+4,17)

- SUONO BIANCO: ADSR con valori da provare:
  forma d' onda POKE S+4,129 (128=spento)

LE NOTE

Ora finalmente, usando una sola voce, proverete a suonare un motivo:
sara' necessario eseguire tutte le operazioni di programmazione gia' viste, ed introdurre un ciclo che legga la nota, cioe' i due dati di frequenza in cui e' divisa, la sua durata e successivamente passi alla nota successiva.
Ogni nota e' definita da una frequenza che viene riprodotta dal COMMODORE 64 dopo aver memorizzato nei registri 0,1 (per la voce 1) 7,8 (per la voce 2) 14,15 (per la voce 3):

il quoziente reso intero ottenuto
dividendo la frequenza stessa per 0.06097.

Se ad esempio volete suonare un suono con frequenza pari a 1500 Hz dovrete memorizzare il numero:

1500 / 0.06097 = 24602 = Fc (Frequenza commodore)

operazione questa che richiedera' ancora un passaggio, cioe' la scissione del DEC 24602 in due DEC da inserire nei due registri:

- nel registro 1 memorizzate la parte intera del quoziente ottenuto dividendo Fc per 256
- nel registro 0 memorizzate il resto (cioe' la differenza tra Fc ed il contenuto del registro 1 moltiplicato per 256):

in reg. 1   INT (Fc/256) = INT (24602/256) = 96
in reg. 0   24602 - (256*96) = 24602 - 24576=26

Percio': POKE S+0,26:POKE S+1,96 (dove S=54272)

Su questo libro troverete i valori gia' tabellati per tutte le note ed un metodo per far calcolare le frequenze direttamente da programma, percio' niente paura ...non e' complicato.

Eseguiamo ora la scala di DO:

NOTA	REG 1	REG 0
DO	33	135
RE	37	162
MI	42	62
FA	44	193
SOL	50	60
LA	56	99
SI	63	75
DO	67	15

Il programma e' il solito:

10-11 azzerare il SID
20   alzare il volume
50   selezionare A-D (31-15=16 percio' A=1,D=15)
60   selezionare S-R (31-15=16 percio' S=1,R=15)
61   ciclo di lettura valori di frequenza e durata della nota e memorizzazione della parte alta della frequenza nel registro 1

62 memorizza la parte bassa della frequenza nel registro 0
70 seleziona l'onda triangolare (comincia l'emissione sonora) ed elabora un ciclo di ritardo per la durata della nota.
80 chiude la forma d'onda (si ferma l'emissione sonora)
81 ciclo di ritardo per staccare una nota dalla successiva, chiusura del ciclo di lettura note e RESTORE dei dati per ricominciare.
90-91-92-93 dati delle note disposto come segue: parte alta frequenza, parte bassa, durata.

```
10 S = 54272
11 FOR K=0 TO 24:POKE S+K,0:NEXT
20 POKE S+24,15
30 REM S+1 ED S VENGONO IMPOSTATE PIU'AVANTI
50 POKE S+5,31
60 POKE S+6,31
61 FOR H=1 TO 8:READ A,B,C:POKE S+1,A
62 POKE S,B:REM FREQUENZA A=ALTA B=BASSA
70 POKE S+4,17:FOR I=1 TO C:NEXT I
80 POKE S+4,16
81 FOR N=1TO50:NEXT N:NEXT H:RESTORE:GOTO 61
90 DATA 33,135,100,37,162,100
91 DATA 42, 62,100,44,193,100
92 DATA 50, 60,100,56, 99,100
93 DATA 63, 75,100,67, 15,100
```

Provate ora ad eseguire la stessa cosa con due voci, le modifiche da apportare sono poche:

50-60 selezionare l'ADSR anche per la seconda voce (registri 12 e 13 valore prescelto per l'esempio = 36 pari ad A=2, D=4, S=2, R=4)
61-62 memorizzare i valori delle frequenze anche nei registri di emissione della voce 2 (7+0 e 7+1)
70-80 accendere e spegnere la forma d'onda (nell'esempio il dente di sega) per la seconda voce.

```
10 S = 54272
11 FOR K=0 TO 24:POKE S+K,0:NEXT
20 POKE S+24,15
50 POKE S+5,31:POKE S+12,36
60 POKE S+6,31:POKE S+13,36
61 FOR H=1 TO 8:READ A,B,C:POKE S+1,A:POKE S+8,A
```

```
62 POKE S,B:POKE S+7,B
70 POKE S+4,17:POKE S+11,33:FOR I=1 TO C:NEXT I
80 POKE S+4,16:POKE S+11,32
81 FOR N=1TO50:NEXT N:NEXT H:RESTORE:GOTO 61
90 DATA 33,135,100,37,162,100
91 DATA 42, 62,100,44,193,100
92 DATA 50, 60,100,56, 99,100
93 DATA 63, 75,100,67, 15,100
```

Se vi piace sperimentare, provate le varie combinazioni dei valori di inviluppo del generatore dei suoni e delle forme d'onda e modificate il programma per avere un'emissione sulla terza voce.

```
10 S = 54272
11 FOR K=0 TO 24:POKE S+K,0:NEXT K
20 POKE S+24,15
30 POKE S+14+3,8:REM ONDA QUADRA SU VOCE 3
50 POKE S+5,31:POKE S+12,36:POKE S+19,6
60 POKE S+6,31:POKE S+13,36:POKE S+20,6
61 FOR H=1 TO 8:READ A,B,C:POKE S+1,A:POKE S+8,A
62 POKE S+15,A:POKE S,B:POKE S+7,B:POKE S+14,B
70 POKE S+4,17:POKE S+11,33:POKE S+18,65
71 FOR I=1 TO C:NEXT I
80 POKE S+4,16:POKE S+11,32:POKE S+18,64
81 FOR N=1TO50:NEXT N:NEXT H:RESTORE:GOTO 61
90 DATA 33,135,100,37,162,100
91 DATA 42, 62,100,44,193,100
92 DATA 50, 60,100,56, 99,100
93 DATA 63, 75,100,67, 15,100
```

Dovreste anche sapere che una nota di uno strumento (quale il solito pianoforte) non e' mai emessa da tre voci perfettamente uguali, anzi esse vengono volutamente stonate tra loro di una piccola entita' in fase di accordatura. Possiamo simulare questo fenomeno modificando il programma in una versione leggermente diversa. Infatti invece di leggere i due valori della frequenza gia' elaborati e pronti per la loro memorizzazione nei registri, si leggera' il valore A della frequenza da suonare con la voce 1 e lo si stonera' per le voci 2 e 3 (A2=A*.99:A3=A*.98) scindendolo poi nelle due parti da memorizzare nei registri come parte alta e parte bassa della frequenza.Questa operazione, tenendo conto che l'istruzione POKE lavora per numeri interi, e' molto semplice:

parte alta POKE S+1,A/256
parte bassa, ossia il resto POKE S,A-INT(A/256)*256.

Si e' anche modificato per i vari registri l' ADSR in modo che le tre voci siano simili:

```
 10 S = 54272
 11 FOR K=0 TO 24:POKE S+K,0:NEXT K
 20 POKE S+24,15
 30 POKE S+14+3,8:REM ONDA QUADRA SU VOCE 3
 50 POKE S+5,31:POKE S+12,31:POKE S+19,31
 60 POKE S+6,41:POKE S+13,41:POKE S+20,41
 61 FOR H=1 TO 8:READ A,C:A2=A*.99:A3=A*.98
 62 POKE S+1,A/256:POKE S+8,A2/256:POK2 S+15,A3/256
 64 POKE S,A-INT(A/256)*256
 65 POKE S+7,A2-INT(A2/256)*256
 66 POKE S+14,A3-INT(A3/256)*256
 70 POKE S+4,17:POKE S+11,33:POKE S+18,65
 71 FOR I=1 TO C:NEXT I
 80 POKE S+4,16:POKE S+11,32:POKE S+18,64
 81 FOR N=1TO50:NEXT N:NEXT H:RESTORE:GOTO 61
 90 DATA  8583,100, 9634,100
 91 DATA 10814,100,11457,100
 92 DATA 12860,100,14435,100
 93 DATA 16203,100,17167,100
```

Nel programma successivo non si fa altro che elaborare la versione del primo programma suonando la scala di DO sia in salita che in discesa.

```
9  FOR I=1 TO 8 READ A(I),B(I),C(I):NEXT
10 S = 54272:P=1:P1=1:P2=8
11 FOR K=0 TO 24:POKE S+K,0:NEXT K
20 POKE S+24,15
50 POKE S+5,31:POKE S+12,36
60 POKE S+6,31:POKE S+13,36
61 FOR H=P1 TO P2 STEP P:POKE S+1,A(H):POKE S+8,A(H)
62 POKE S,B(H):POKE S+7,B(H)
70 POKE S+4,17:POKE S+11,33:FOR I=1 TO C(H):NEXT C
80 POKE S+4,16:POKE S+11,32
81 FOR N=1TO50:NEXT N:NEXT H
82 P=(-P):IF P=-1 THEN P1=8:P2=1:GOTO61
83 P1=1:P2=8:GOTO61
90 DATA 33,135,100,37,162,100
91 DATA 42, 62,100,44,193,100
92 DATA 50, 60,100,56, 99,100
93 DATA 63, 75,100,67, 15,100
```

La stessa cosa viene ora fatta sfasando le due voci (la prima sale mentre la seconda scende e viceversa:

bastera' modificare le due righe:

```
61  FOR H=P1 TO P2 STEP P:POKE S+1,A(H):POKE S+8,A(9-H)

62 POKE S,B(H):POKE S+7,B(9-H):REM 9-H !!!!!!
```

Il prossimo programma produce arpeggi in tonalita' di DO maggiore (le note sono DO - MI - SOL sulle varie ottave:

```
8  FOR I=1 TO 8 READ A(I),B(I):NEXT
9  FOR I=1 TO 8 READ A1(I),B1(I):NEXT
10 S = 54272:P=1:P1=1:P2=7
11 FOR K=0 TO 24:POKE S+K,0:NEXT K
20 POKE S+24,15
50 POKE S+5,31:POKE S+12,31
60 POKE S+6,147:POKE S+13,147
61 FOR H=P1 TO P2 STEP P:POKE S+1,A(H):POKE S+8,A1(8-H)
62 POKE S,B(H):POKE S+7,B1(8-H)
70 POKE S+4,17:POKE S+11,33:FOR I=1 TO 100:NEXT I
80 POKE S+4,16:POKE S+11,32
81 FOR N=1TO50:NEXT N:NEXT H
82 P=(-P):IF P=-1 THEN P1=7:P2=1:GOTO61
83 P1=1:P2=7:GOTO61
90 DATA  8, 97, 10,143, 12,143
91 DATA 16,195, 21, 31, 25, 30, 33,135
92 DATA 33,135, 42, 62, 50, 60
93 DATA 67, 15, 84,125,100,121,134,130
```

Usando le stesse note contemporaneamente si puo' creare un accordo od una serie di accordi:

```
9  FOR I=1 TO 9 READ A1(I),B1(I):NEXT
10 S = 54272
11 FOR K=0 TO 24:POKE S+K,0:NEXT K
20 POKE S+24,15
50 POKE S+5,31:POKE S+12,31:POKE S+19,31
60 POKE S+6,31:POKE S+13,31:POKE S+20,31
61 READ IX:IF IX=-1 THEN 100
63 POKE S+1,A(IX*3+1):POKE S+8,A(IX*3+2)
64 POKE S+15,A(IX*3+3)
65 POKE S,B(IX*3+1):POKE S+7,B(IX*3+2)
66 POKE S+14,B(IX*3+3)
70 POKE S+4,17:POKE S+11,17:POKE S+18,17
71 FOR I=1 TO 400:NEXT I
80 POKE S+4,16:POKE S+11,16:POKE S+18,16
81 FOR N=1TO 200:NEXT N:GOTO61
```

```
89 REM    MI      SOL     DO
90 DATA 42, 62,  50,60,  67,15
91 REM    RE      SOL     SI
92 DATA 37,162,  50,60,  63,75
93 REM    LA       DO     FA
94 DATA 56, 99,  67,15,  89,131
95 REM 0=DO 1=SOL 2=FA (ACCORDI)
96 DATA 0,1,0,2,0,1,0,0,0,-1
100 FOR K=0 TO 24:POKE S+K,0:NEXT K:END
```

Per poter aggiungere alcune regole in merito alla composizione degli arpeggi o degli accordi e' bene far mente locale alla tastiera di un pianoforte o di un organo: questa ha uno schema ripetitivo fatto da sette tasti bianchi e da cinque tasti neri (questi ultimi disposti in due gruppi di due e di tre). La differenza di tonalita' tra un tasto ed il successivo, comprendendo anche i tasti neri, e' chiamato semitono, percio' tra una nota e quella di un'ottava superiore o inferiore esistono 7 + 5 = 12 semitoni. Il tasto bianco precedente alla coppia di tasti neri e' il DO.

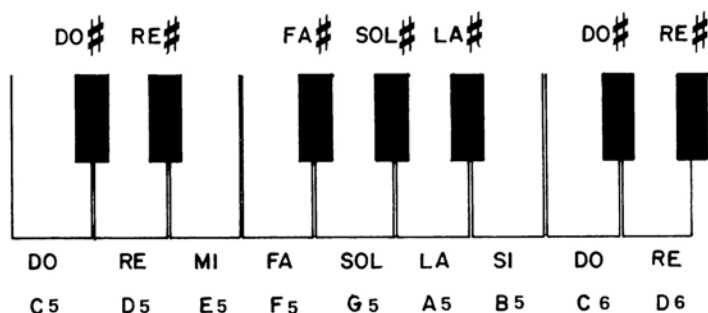



FIG.25. Schema di tastiera di pianoforte

In tema di matematica musicale la cosa si risolve molto semplicemente:

- la frequenza di una nota e' esattamente la meta' della frequenza della nota superiore di un'ottava

- la frequenza di una nota e' il doppio di quella della stessa nota inferiore di un'ottava

- all'interno di un'ottava ogni semitono viene
  ricavato dal precedente moltiplicandone la
  frequenza per la radice dodicesima di 2 cioe'
  per 1,05946309.....

Se ad esempio il LA della quarta ottava (LA-4) ha frequenza 7217 Hz, il LA-3 avra' frequenza la meta', cioe' 3608; mentre il LA# 4 si ottiene col prodotto:
7217*1,05946309=7646.

Se consultate la tabella di altri manuali troverete le note con una denominazione diversa da quella usuale per noi italiani, la corrispondenza e' quella riportata nello schemino della tastiera del pianoforte,facendo attenzione che il LA viene preso come inizio di numerazione alfabetica delle note ma il numero dell' ottava cambia sempre sul DO (potra' sembrarvi un controsenso ma A-4,cioe' il LA dell' ottava num.4 ha un suono piu' acuto di G-4 cioe' SOL della stessa ottava. Questo metodo di chiamare le note e' d'altra parte molto comodo per il calcolatore quando, tra poco, si introdurra' un criterio che consenta di dare come dati le stesse note al posto delle corrispondenti frequenze.

Se volete comporre un arpeggio od un accordo in maggiore dovrete suonare le seguenti note:
- nota dell'accordo
- nota superiore di 4 semitoni (o inferiore di 8)
- nota superiore di 7 semitoni (o inferiore di 5)

Se l'accordo e' in minore:
- nota dell'accordo
- nota superiore di 3 semitoni (o inferiore di 9)
- nota superiore di 7 semitoni (o inferiore di 5)

Se l'accordo e' di settima maggiore:
- nota dell'accordo
- nota superiore di 5 semitoni (o inferiore di 7)
- nota superiore di 7 semitoni (o inferiore di 5)
- nota superiore di 10 semitoni (o inferiore di 2)

questo accordo ha **bisogno** di 4 voci per essere suonato completamente, percio' dovrete decidere le tre note da suonare.

Se l'accordo e' di settima maggiore:
- nota dell'accordo
- nota superiore di 5 semitoni (o inferiore di 7)
- nota superiore di 7 semitoni (o inferiore di 5)
- nota superiore di 10 semitoni (o inferiore di 2)

questo accordo ha bisogno di 4 voci per essere suonato completamente, percio' dovrete decidere le tre note da suonare.

SUONARE CON LE TRE VOCI.

Esiste una difficolta' suonando con le tre voci derivante dalla relativa lentezza dal Basic nell'espletare le istruzioni del programma, il suono delle tre voci risulta cosi' non contemporaneo. Cio' comporta la necessita' in fase di programmazione di evitare al massimo istruzioni che provochino perdite di tempo o rallentamenti o che possano variare le durate del suono. Uno dei metodi per ovviare a questi inconvenienti senza ricorrere al linguaggio macchina (che in questo caso sarebbe l'ideale), e' quello di memorizzare tutte le note all'interno di matrici, in modo da non perdere tempo nelle letture o nei calcoli durante il ciclo dedicato al suono.
Inoltre ogni nota sara' dotata di una sua durata (numero intero) ed una sequenza di IF controllera' se la sua durata e' gia' ultimata o meno. Anche le istruzioni di IF devono essere studiate in modo da impegnare ugual tempo sia nel caso di esecuzione che in caso contrario (altrimenti potete dire addio al ritmo.)
Nel programma seguente si e' anche provveduto a generare automaticamente le frequenze delle note:

- in primo luogo si genera l'ottava num. 7 sulla base della frequenza del DO-7 (detto C-7 con l'altra nomenclatura),
- ogni nota viene ricavata dalla corrispondente della ottava num. 7 dimezzandone la frequenza un numero di volte pari alla differenza tra la sua ottava e la ottava num. 7.
- la frequenza F cosi' ricavata viene divisa in due parti: A (alta) e B (bassa) e memorizzata in due matrici A (V,NO) e B (V,NO) dove il primo indice tiene conto delle tre voci, il secondo del numero d'ordine della nota.

Sempre per accelerare il ciclo si memorizza l'indirizzo dei registri di frequenza e quello delle forme d'onda in una matrice FO (V,F) in cui il primo indice e' quello relativo alla voce ed il secondo e' relativo a:

- 0   locazione registro Forma d'Onda (4, 7+4, 14+4)
- 1   valore della Formà d'Onda (accesa 17, 33, 65)
- 2   valore della Forma d'Onda (spenta 16, 32, 64)

Il ciclo di suono della nota viene effettuato decrementando per ogni nota il valore della sua durata (memorizzato in una matrice DU (DV)) e passando alla nota successiva quando la durata e' scesa a zero. Naturalmente se la durata e' ancora positiva il suono

della nota viene riacceso (notate che per non introdurre variazioni di tempo tra la nota di una voce e quella di un'altra si memorizza la nota nei registri tutte le volte; se invece la durata e' scesa a zero si passa alla nota successiva (e se in questa A=0 vuol dire che la nota non suona) si carica la sua durata e si continua alla voce successiva:

```
1 REM V = VOCI = 3  NOTE = 100 MAX
2 REM A(3,100) PARTE ALTA  DELLA FREQUENZA
3 REM B(3,100) PARTE BASSA DELLA FREQUENZA
4 REM DU(3,100) DURATA DI OGNI NOTA PER LE TRE VOCI
5 REM FO(3,3) FORMA D'ONDA 0= LOCAZ. REGISTRO
6 REM                      1= ONDA ON : 17 33 65
7 REM                      2= ONDA ON : 16 32 64
8 REM FR(12) FREQUENZA SEMITONI OTTAVA NUM.7
9 REM NT$ NOTA IN NOTAZIONE ANGLOSASSONE (A,A#,B,C,C#...)
10 REM     SEGUITA DALL' OTTAVA (ES A#4,B5...)
11 REM NN(3) NUMERO DI NOTE PER OGNI VOCE (ULTIMA=*)
12 REM CN(3) CONTATORE NOTA CHE STA SUONANDO
13 REM RF(3,2) REGISTRI DOVE MEMORIZZARE LE FREQUENZE
15 REM ************************************************
50 REM DIMENSIONAMENTO MATRICI
51 DIM A(3,300),B(3,300),DU(3,300),FO(3,3),FR(12),CN(13)
52 DIM NN(3),RF(3,2)
99 REM AZZERAMENTO DEL SID
100 S= 54272:FORK=0TO24:POKES+K,0:NEXTK
110 V=0
119 REM MEMORIZZAZIONE DATI NEI REGISTRI
120 FOR I= 0 TO 6 :READ P
130 POKE S+7*V+I,P:NEXT I
131 V= V+1: IF V<3 THEN120
133 DATA 0,0,0,8,0,24,96
134 DATA 0,0,0,8,0,25,128
135 DATA 0,0,0,8,0,21,128
136 REM 0,8 DATI ONDA QUADRA, 24,250 = A,D,S,R
137 FOR V=1 TO 3:READ FO(V,1),FO(V,2):NEXT V
138 DATA 65,64,65,64,65,64
139 FO(1,0)=S+4:FO(2,0)=S+4+7:FO(3,0)=S+4+14
140 REM CICLO PREPARAZIONE OTTAVA NUM.7
143 FOR F=0TO11
145 FR(F)=57743*(2↑(F/12))
147 NEXT F
150 REM PREPARAZIONE REGISTRI FREQUENZE
155 FOR V= 1TO3: FORJ =0TO1:RF(V,J)=S+J+(V-1)*7:NEXTJ:NEXTV
200 REM LETTURA NOTE, TRASFORMAZIONE IN
210 REM FREQUENZE E MEMORIZZAZIONE IN A(V,CN),B(V,CN)
220 FOR V=1TO3
230 REM DATI VOCE NUMERO V
239 I= 1
```

```
240 READ NT$,DU(V,I):
250 IF NT$="*" THEN NN(V)=I:GOTO330:REM FINE NOTE VOCE
255 REM SE NT$ VALE S LA VOCE FA SILENZIO
260 IF NT$="S"THEN A(V,I)=0:B(V,I)=0:GOTO320
270 OTTAVA=VAL(RIGHT$(NT$,1)):REM NUMERO A DESTRA = OTTAVA
275 REM FREQUENZA DELLA NOTA IN OTTAVA NUM.7
280 F= ASC(LEFT$(NT$,1))-65
281 ON F+1 GOSUB 5000,5010,5020,5030,5040,5050,5060
285 REM SE C'E' IL # SI PASSA AL SEMITONO SUPERIORE
290 IF MID$(NT$,2,1)="#" THEN F=F+1
295 F1=FR(F)
300 F1=F1/(2↑(7-OTTAVA)):REM FREQUENZA SULL' OTTAVA GIUSTA
310 A(V,I)=INT(F1/256):B(V,I)=F1-A(V,I)*256
320 I=I+1:GOTO240
330 NEXT V
399 REM ************************************************
400 REM CICLO SONORO SULLE TRE VOCI
401 REM 410 VOLUME
402 REM 425 DECREMENTO DURATA
403 REM 430 TEST PER FINE DURATA
404 REM 450 SE FREQ DIVERSA DA 0 FORMA ONDA ACCESA
405 REM 460 IMPOSTA LA FREQUENZA
406 REM 470 TEST FINE NOTE PER LE TRE VOCI
407 REM ************************************************
410 EN=0:POKE S+24,15
420 FOR V= 1TO3
425 DU(V,CN(V))=DU(V,CN(V))-1
430 IFDU(V,CN(V))<1THENPOKEFO(V,0),FO(V,2):CN(V)=CN(V)+1
440 A=A(V,CN(V)):B=B(V,CN(V))
450 IF A=<0 THEN POKEFO(V,0),FO(V,2):POKERF(V,1),
    A:POKERF(V,0),B
451 IF A>0 THEN POKEFO(V,0),FO(V,1):POKERF(V,1),
    A:POKERF(V,0),B
460 REM      F(V,1),A:POKERF(V,0),B
470 IF CN(V)=NN(V) THEN EN=EN+1:POKEFO(V,0),FO(V,2):
    IF EN=3THEN END
480 NEXT V
490 GOTO 420
999 REM        DATI PRIMA VOCE
1000 DATA G#3,2,C#4,2,E4,2,G#3,2,C#4,2,E4,2
1010 DATA G#3,2,C#4,2,E4,2,G#3,2,C#4,2,E4,2
1020 DATA G#3,2,C#4,2,E4,2,G#3,2,C#4,2,E4,2
1030 DATA G#3,2,C#4,2,E4,2,G#3,2,C#4,2,E4,2
1040 DATA  A3,2,C#4,2,E4,2, A3,2,C#4,2,E4,2
1050 DATA  A3,2, D4,2,F#4,2, A3,2, D4,2,F#4,2
1060 DATA G#3,2,C4,2,F#4,2,G#3,2,C#4,2,E4,2
1070 DATA G#3,2,C#4,2,D#4,2,F#3,2,C4,2,D#4,2
1080 DATA E3,2,G#3,2,C#4,2,G#3,2,C#4,2,E4,2
1090 DATA G#3,2,C#4,2,E4,2,G#3,2,C#4,2,E4,2
1100 DATA G#3,2,D#4,2,F#4,2,G#3,2,D#4,2,F#4,2
```

```
1110 DATA G#3,2,D#4,2,F#4,2,G#3,2,D#4,2,F#4,2
1120 DATA G#3,2,C#4,2,E4,2,G#3,2,C#4,2,E4,2
1130 DATA A3,2,C#4,2,F#4,2,A3,2,C#4,2,F#4,2
1140 DATA G#3,2,B3,2,E4,2,G#3,2,B3,2,E4,2
1150 DATA A3,2, B3,2,D#4,2,A3,2,B3,2,D#4,2
1160 DATA G#3,2,B3,2,E4,2,G#3,2,B3,2,E4,2
1170 DATA G#3,2,B3,2,E4,2,G#3,2,B3,2,E4,2
1180 DATA G3,2,B3,2,E4,2,G3,2,B3,2,E4,2
1190 DATA G3,2,B3,2,E4,2,G3,2,B3,2,E4,2
1200 DATA G3,2,B3,2,F4,2,G3,2,B3,2,F4,2
1210 DATA G3,2,B3,2,F4,2,G3,2,B3,2,F4,2
1220 DATA G3,2,C4,2,E4,2,G3,2,B3,2,E4,2
1230 DATA G3,2,A#3,2,E4,2,F#3,2,A#3,2,E4,2
1240 DATA F#3,2,B3,2,D4,2,F#3,2,B3,2,D4,2
1250 DATA G3,2,B3,2,C#4,2,E3,2,B3,2,C#4,2
1260 DATA F#3,2,B3,2,D4,2,F#3,2,B3,2,D4,2
1270 DATA F#3,2,A#3,2,C#4,2,F#3,2,A#3,2,C#4,2
1280 DATA B3,2,D4,2,F#4,2,B3,2,D4,2,F#4,2
1290 DATA B3,2,D#4,2,F#4,2,B3,2,D#4,2,F#4,2
1300 DATA *,0
1999 REM       DATI SECONDA VOCE
2000 DATA C#3,24,B2,24
2010 DATA  A2,12,F#2,12,G#2,12,G#2,12
2020 DATA C#3,24,C2,24,C#3,12,F#2,12
2030 DATA B2,12,B2,12,E3,24,E3,24
2040 DATA D3,24,C3,6,B2,6,A#2,12
2050 DATA B2,12,E2,6,G2,6,F#2,12
2060 DATA F#1,12,B1,24
2200 DATA *,0
2999 REM       DATI TERZA VOCE
3000 DATA C#2,24,B1,24
3010 DATA  A1,12,F#1,12,G#1,12,G#1,12
3020 DATA G#2,18,G#4,5,G#4,1,G#4,18
3030 DATA G#4,5,G#4,1,G#4,12,A4,12
3040 DATA G#4,12,F#4,6,B4,6
3050 DATA E4,6,S,18,E1,18,G4,5,G4,1
3060 DATA G4,18,G4,5,G4,1,G4,18,F#4,6
3070 DATA F#4,12,G4,6,E4,6,F#4,12,F#4,12
3080 DATA B2,24
3200 DATA *,0
5000 F=9:RETURN
5010 F=11:RETURN
5020 F=0:RETURN
5030 F=2:RETURN
5040 F=4:RETURN
5050 F=5:RETURN
5060 F=7:RETURN
```

COME PASSARE DAL RIGO MUSICALE AI DATI.

Il brano, tratto dal primo movimento della Sonata quasi una Fantasia opera 27 N 2 di Ludwig Van Beethoven, e' stato semplificato in modo da suonare le 3 voci possibili (escludendo cosi' purtroppo alcune note) e tenendo presenti le seguenti definizioni musicali:

alle semicrome e' stato assegnato il valore di tempo 1

alle crome il doppio

alle semiminime (in questo caso pari a tre crome) il valore 6

alle minime il valore 12

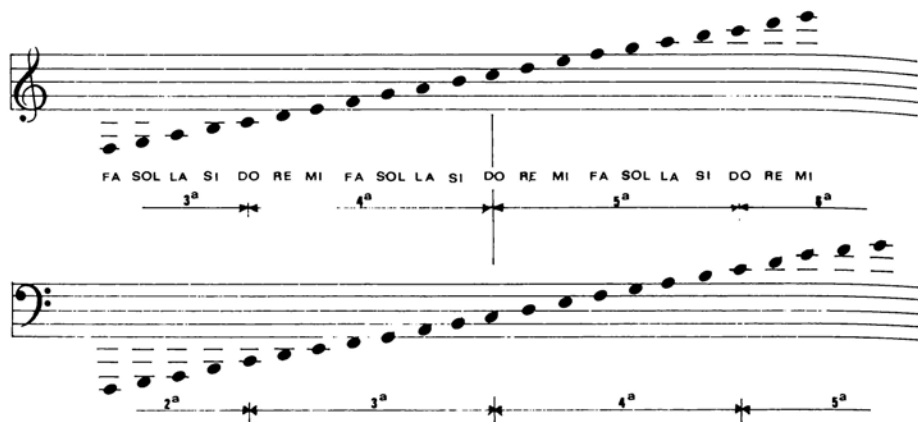
alle semibrevi il valore 24

Inoltre la terza voce esegue in parte l'accompagnamento, e in parte la parte principale del canto.

Vi raccomandiamo, prima di lanciarvi nel RUN fatidico, di salvare il programma (se per caso vi dimenticaste di scrivere POKE S + n.di registro e scriveste semplicemente POKE n. di registro, otterreste pasticci in pagina 0 e il COMMODORE 64 non risponderebbe piu' coerentemente ai vostri comandi, impedendovi quindi anche un'operazione di salvataggio).

Inoltre controllate bene piu' volte tutti i dati, in quanto basta un errore di tempo o di note per rovinare tutto.

Per rendere piu' comprensibile il passaggio dalle note segnate sul pentagramma ai dati delle 3 voci, sono state segnate di seguito le nomenclature delle note sul pentagramma. Infatti la musica viene scritta (da chi di musica ne sa sicuramente piu' dei sottoscritti e quindi

vorra' perdonare certe semplificazioni cosi' drastiche ma del resto necessarie per non perdere di vista l'obbiettivo finale) con l'utilizzo di 5 righe parallele che servono di supporto alle note (queste ultime saranno a cavallo di una riga o perfettamente contenute nello spazio tra due righe).

Sara' necessario interpretare, attraverso la posizione e la forma, il tipo di nota, l'ottava e la durata ed inoltre decidere quali note far suonare.

Quest'ultimo passaggio e' stato eseguito permettendovi di leggere ogni voce in modo separato dalle altre.
Quando sarete diventati pratici, potrete eseguire l'operazione piu' complessa, ossia partire direttamente dalla musica scritta per i musicisti, i quali (beati loro!...) hanno a disposizione strumenti molto piu' efficaci delle 3 voci del COMMODORE 64 e sono altresi' allenati a leggere le note e a trasformarle in interpretazioni, che, pur amando l'elettronica, riteniamo siano tuttora inimitabili.

FIG. 26. Note suonate dalla prima voce

FIG. 27. Note suonate dalla seconda voce

FIG.28. Note suonate dalla terza voce
si alternano bassi ed acuti

USO DEI FILTRI E RISONANZA.

Si e' visto in precedenza come le forme d'onda contengano una mescolanza di frequenze dette armoniche superiori e tutte multiple della fondamentale. Il COMMODORE 64 ha la possibilita' di tagliare determinate frequenze in modo che non vengano emesse, cio' puo' essere fatto in tre modi:

- filtro passa basso: lascia passare solo le frequenze inferiori a quella assegnata (che viene detta frequenza di taglio)

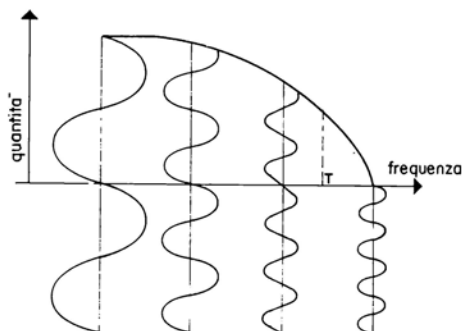



FIG.29.

- filtro passa alto: lascia passare solo le frequenze superiori alla frequenza assegnata

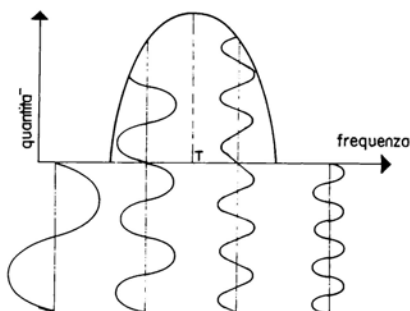



FIG.30.

- filtro passa banda: lascia passare solo la frequenza intorno a quella di taglio.

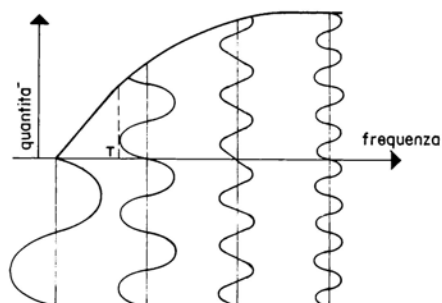



FIG. 31.

La frequenza di taglio deve essere impostata nei registri 21 e 22 facendo attenzione che nel registro 21 si usano solo I TRE PRIMI Bit (i numeri 0,1,2) e questi impostano la parte bassa della frequenza di taglio; mentre nel registro 22 vengono utilizzati tutti gli 8 Bit per impostare la parte alta della frequenza di taglio.
Una volta definita la frequenza bisogna "accendere" il filtro voluto utilizzando i Bit num. 4,5,6 del registro 24 che avete gia' usato per il volume (BIt 0,1,2,3):

- Bit 4, quando e' a 1 si accende il filtro passa basso
- Bit 5, quando e' a 1 si accende il filtro passa banda
- Bit 6, quando e' a 1 si accende il filtro passa alto
- Bit 7, non modifica i filtri ma accende o spegne la voce num. 3 e questo e' utile per effetti speciali.

Se ad esempio volete accendere il filtro passa banda dovrete memorizzare nel registro 24 il num. 215+15=47, per il filtro passa basso 214+15=31, per il passa alto 216+15=79.
Inoltre e' anche possibile ottenere il funzionamento contemporaneo dei registri: usando ad esempio il passa alto ed il passa basso otterrete l'effetto di inibire solamente le frequenze adiacenti a quella di taglio (ossia il contrario del risultato ottenuto col filtro

passa banda) cio' verra' eseguito memorizzando 216+214+15=95 sempre nel registro 24.

filtri passa alto e passa basso combinati

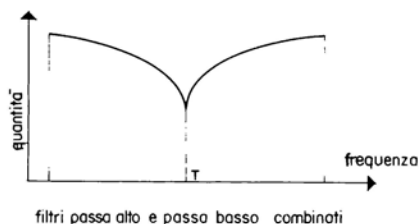
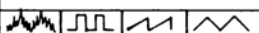
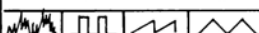
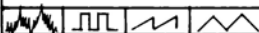



FIG.32.

Perche' il filtro cominci effettivamente a funzionare e' necessario decidere su quale voce deve essere applicato: cio' viene eseguito con il registro 23:

- Bit 0, quando viene i;postato a 1 abilita la filtratura sulla voce 1
- Bit 1, abilita la filtratura della voce 2
- Bit 2, abilita la filtratura della voce 3
- Bit 3, filtra eventuali ingressi audio-esterni
- Bit 4-5-6-7, controllano l'effetto di risonanza del filtro (valori da 1 a 15) praticamente potenziando tale frequenza.

L'uso dei filtri e' abbastanza complesso e vario cosi' da rendere difficile una schematizzazione, in pratica un uso accurato consente l'ottenimento della forma d'onda desiderata partendo da onde di forma diversa.
Provate ad esempio il programma seguente per osservare l'effetto dei filtri sull'emissione sonora di una stessa nota.

```
10 S = 54272
11 FOR K=0 TO 24:POKE S+K,0:NEXT K
12 POKE S+21,0:POKE S+22,80:REM FREQUENZA DI TAGLIO
13 POKE S+23,1:REM ACCENDE FILTRI VOCE 1
30 POKE S+1,16:POKE S,195:REM NOTA DO-4
50 POKE S+5,15:REM A-D
60 POKE S+6,15:REM S-R
61 POKE S+2,0:POKE S+3,8:REM DATI ONDA QUADRA
70 POKE S+24,15:GOSUB 100:REM SENZA FILTRO
71 POKE S+24,31:GOSUB 100:REM FILTRO PASSA BASSO
72 POKE S+24,47:GOSUB 100:REM FILTRO PASSA BANDA
73 POKE S+24,79:GOSUB 100:REM FILTRO PASSA ALTO
74 POKE S+24,95:GOSUB 100:GOTO60:REM FILTRI ALTO+BASSO
100 POKE S+4,65:FOR I=1 TO 200:NEXT I
110 POKE S+4,64:FOR I=1 TO 100:NEXT I:RETURN
```

Modificate il programma in modo da variare il tipo di onda:

```
100 POKE S+4,17:FOR I=1 TO 200:NEXT I
110 POKE S+4,16:FOR I=1 TO 100:NEXT I:RETURN
```

FIG.33. SCHEMA RIASSUNTIVO DELLE LOCAZIONI DI MEMORIA DEL SID E DELLE LORO FUNZIONI

REG+ S	VO.	7	6	5	4	3	2	1	0	FUNZ.	R&W
0	V1	PARTE ALTA DELLA FREQUENZA DELLA NOTA									W
1	V1	PARTE BASSA DELLA FREQUENZA DELLA NOTA									W
2	V1	PARTE BASSA AMPIEZZA ONDA QUADRA									W
3	V1	-	-	-	-	P.ALTA ONDA QUAD					W
4	V1					TES	R.M	SYN	USC	ONDA	W
5	V1	PARTE ATTACK				PARTE DEKAY				GEN.	W
6	V1	PARTE SUSTAIN				PARTE RELEASE				INV.	W
		7	6	5	4	3	2	1	0		
7+0	V2	PARTE ALTA DELLA FREQUENZA DELLA NOTA									W
7+1	V2	PARTE BASSA DELLA FREQUENZA DELLA NOTA									W
7+2	V2	PARTE BASSA AMPIEZZA ONDA QUADRA									W
7+3	V2	-	-	-	-	P.ALTA ONDA QUAD					W
7+4	V2					TES	R.M	SYN	USC	ONDA	W
7+5	V2	PARTE ATTACK				PARTE DEKAY				GEN.	W
7+6	V2	PARTE SUSTAIN				PARTE RELEASE				INV.	W
		7	6	5	4	3	2	1	0		
14+0	V3	PARTE ALTA DELLA FREQUENZA DELLA NOTA									W
14+1	V3	PARTE BASSA DELLA FREQUENZA DELLA NOTA									W
14+2	V3	PARTE BASSA AMPIEZZA ONDA QUADRA									W
14+3	V3	-	-	-	-	P.ALTA ONDA QUAD					W
14+4	V3					TES	R.M	SYN	USC	ONDA	W
14+5	V3	PARTE ATTACK				PARTE DEKAY				GEN.	W
14+6	V3	PARTE SUSTAIN				PARTE RELEASE				INV.	W
		7	6	5	4	3	2	1	0		
21		-	-	-	-	-	FREQ. BASSA DEL FILTRO				W
22		FREQUENZA PARTE ALTA DA UTILIZZ COL FILTRO									W
23		CONTR. RISONANZ				FEX	FV3	FV2	FV1	RIS.	W
24		NO V.3	FILTRI ACC. P.A	PBN	P.B	REGOL.VOLUME DA 1 A 15				VOL+ON FILTRI	W
25		USATO DA MICROPROCESSORE									R
26		USATO DAL MICROPROCESSORE									R
27		RIPETE BIT ALTI OSCILL. VOCE 3								EFF.	R
28		RIPETE USCITA GEN.INVIL.VOCE 3								SPEC	R

e poi ancora con 33 e 32 al posto di 17,16.
Variate anche la frequenza di taglio:

```
12 POKE S+21,7:POKE S+22,160:REM FREQUENZA DI TAGLIO
```

FREQUENZE DELLE NOTE MUSICALI

Si riportano in tabella, per le note musicali relative a 8 ottave, le frequenze ed i corrispondenti valori decimali da impostare nella coppia di registri (0,1 per la voce 1 ; 7,8 per la voce 2 ; 14,15 per la voce 3). Tali valori verranno inseriti con l' utilizzo della funzione POKE.

NOTA MUSICALE		FREQUENZA			NOTA MUSICALE		FREQUENZA		
NOTA	OTTAVA	DEC.	BIT ALTO	BIT BASSO	NOTA	OTTAVA	DEC.	BIT ALTO	BIT BASSO
DO	0	268	1	12	DO	4	4291	16	195
DO#	0	284	1	28	DO#	4	4547	17	195
RE	0	301	1	45	RE	4	4817	18	209
RE#	0	318	1	62	RE#	4	5103	19	239
MI	0	337	1	81	MI	4	5407	21	31
FA	0	358	1	102	FA	4	5728	22	96
FA#	0	379	1	123	FA#	4	6069	23	181
SOL	0	401	1	145	SOL	4	6430	25	30
SOL#	0	425	1	169	SOL#	4	6812	26	156
LA	0	451	1	195	LA	4	7217	28	49
LA#	0	477	1	221	LA#	4	7647	29	223
SI	0	506	1	250	SI	4	8101	31	165
DO	1	536	2	24	DO	5	8583	33	135
DO#	1	568	2	56	DO #	5	9094	35	134
RE	1	602	2	90	RE	5	9634	37	162
RE#	1	637	2	125	RE #	5	10207	39	223
MI	1	675	2	163	MI	5	10814	42	62
FA	1	716	2	204	FA	5	11457	44	193
FA#	1	758	2	246	FA#	5	12139	47	107
SOL	1	803	3	35	SOL	5	12860	50	60
SOL#	1	851	3	83	SOL#	5	13625	53	57
LA	1	902	3	134	LA	5	14435	56	99
LA#	1	955	3	187	LA#	5	15294	59	190
SI	1	1012	3	244	SI	5	16203	63	75
DO	2	1072	4	48	DO	6	17167	67	15
DO#	2	1136	4	112	DO#	6	18188	71	12
RE	2	1204	4	180	RE	6	19269	75	69
RE#	2	1275	4	251	RE#	6	20415	79	191
MI	2	1351	5	71	MI	6	21629	84	125
FA	2	1432	5	152	FA	6	22915	89	131
FA#	2	1517	5	237	FA#	6	24278	94	214
SOL	2	1607	6	71	SOL	6	25721	100	121
SOL#	2	1703	6	167	SOL#	6	27251	106	115
LA	2	1804	7	12	LA	6	28871	112	199
LA#	2	1911	7	119	LA#	6	30588	119	124
SI	2	2025	7	233	SI	6	32407	126	151
DO	3	2145	8	97	DO	7	34334	134	30
DO#	3	2273	8	225	DO#	7	36376	142	24
RE	3	2408	9	104	RE	7	38539	150	139
RE#	3	2551	9	247	RE#	7	40830	159	126
MI	3	2703	10	143	MI	7	43258	168	250
FA	3	2864	11	48	FA	7	45830	179	6
FA#	3	3034	11	218	FA#	7	48556	189	172
SOL	3	3215	12	143	SOL	7	51443	200	243
SOL#	3	3406	13	78	SOL#	7	54502	212	230
LA	3	3608	14	24	LA	7	57743	225	413
LA#	3	3823	14	239	LA#	7	61176	238	248
SI	3	4050	15	210	SI	7	64814	253	46

ULTIME NOTIZIE

Se al posto delle forme d' onda gia' ripetutamente usate impostate il valore 129 otterrete un fruscio comunemente chiamato RUMORE BIANCO il suo utilizzo e' fondamentale negli effetti sonori di molti giochi.

Potete decidervi anche a studiare l' assembler per aprire nuovi orizzonti sonori!!!

potete ottenere effetti particolari sommando le frequenze di due voci che lavorano con forme d'onda diverse; in tal caso una delle due voci sara' la terza che non comparira' direttamente (la sua emissione viene inibita come gia' visto portando a 1 il Bit 7 del registro 24) ma la sua frequenza verra' letta nel registro 27 che conterra' valori da 0 a 255 (se non c'avete pensato questa e' una sorgente favolosa di numeri casuali!).
Potete sempre sommare quest'ultima frequenza a quella del filtro creando un filtro variabile, ma col Basic l'effetto sara' poco efficace
Potete modificare il volume durante il suono di una nota abbassandolo e poi alzandolo ancora in modo da simulare un'eco.
Potete mescolare rapidamente frequenze molto differenti per ottenere effetti speciali.

CAPITOLO 6

UTILIZZIAMO LE PERIFERICHE

GESTIONE DEI FILES SU CASSETTE

Per prima cosa sara' bene chiarire che cosa si intende per FILE:

Si tratta di un insieme di DATI di qualsiasi tipo, cioe' numerici, alfanumerici programmi, comandi generalmente ordinati secondo un certo criterio ed archiviati su di un supporto.
Questa definizione a prima vista puo' sembrare poco chiara in quanto esistono svariati tipi di files; chiariamo il concetto con alcuni esempi:

- la vostra rubrica telefonica, contenente i nomi ed i numeri di telefono di tutti i vostri amici, e' un classico esempio di file di dati ordinati per ordine alfabetico ed aventi come supporto la carta

- il calendario e' un altro esempio di file contenente una serie di dati in sequenza

- l'ultimo programma che avete battuto sul vostro COMMODORE 64 e' un file programma.

Si evidenzia quindi che il file non e' nient'altro che un qualsiasi insieme di dati.
Se volete scrivere una serie di dati sul vostro registratore dovete utilizzare alcune istruzioni che vengono di seguito esposte:

A - la prima operazione da farsi e' quella di assegnare un nome alla vostra serie di dati che d'ora in poi chiameremo file: cio' vi consentira' di distinguerlo da altri files.
Per dare inizio alla procedura si utilizza l'istruzione OPEN che informa il vostro COMMODORE 64 dell'intenzione di comunicare con una periferica: il registratore.

La sintassi da usare e' la seguente:

OPEN 3,1,2,"NOME FILE"

dove:

3 e' il numero del canale che volete associare al nome del file (che si chiama NOME FILE) questo numero puo' variare a vostro piacimento da 0 a 255, (ma e' bene che non superiate il num. 127)

1 e' il numero della periferica, il registratore infatti ha il num. 1 come definizione datagli dal sistema

2 e' l'indirizzo secondario che avverte il registratore di prepararsi a REGISTRARE dati; in particolare il valore dell'indirizzo secondario puo' essere:

0 predispone il registratore per leggere dati dal nastro (in questo caso puo' anche essere omesso)
1 predispone il registratore per scrivere i dati; alla fine della registrazione il COMMODORE 64 registra un segnale di fine dati
2 sempre per registrare dati, ma oltre al segnale di fine dati viene registrato anche un egnale di fine nastro, che trova utilizzo quando e' necessario portarsi alla fine della parte di nastro registrato.

NOME FILE e' il nome che avete dato al file e puo'essere una stringa di caratteri lunga un massimo di 16 (senza virgole asterischi punti interrogativi)

esempio: PAOLA e' un nome valido di file
FERRARI 12 anche
SUPERCALIFRAGILISTICHESPIRALIDOSO no!

Ora il COMMODORE 64 in risposta alla OPEN, richiede la pressione del tasto PLAY sul registratore, se avete aperto il file per lettura (modo 0), oppure la pressione dei tasti PLAY & RECORD se avete aperto il file per scrittura (modi 1 - 2)
Da questo momento in poi tutti i dati scambiati tra il registratore ed il COMMODORE 64 vengono memorizzati o letti sul nastro in modo sequenziale, cioe' l'uno di seguito all'altro. Questo modo di registrazione genera un file chiamato FILE SEQUENZIALE ed e' l'unico (oltre al file programma) gestibile dal registratore a cassetta.

B - Dopo aver informato il registratore delle vostre intenzioni di scrivere dei dati, potete passare alla

fase successiva: LA SCRITTURA DEI DATI avviene mediante il comando PRINT# con la seguente sintassi:

```
PRINT#3,"DATO1"
PRINT#3,A
PRINT#3,13
```

Il COMMODORE 64 dirottera' le istruzioni di PRINT anziche' sul video al file specificato dal num. 3 che e' stato associato al nome del file (dalla OPEN) e quindi registrera' sul nastro:

DATO1
il valore della variabile A
13 (come numero)

separandoli con un carattere di RETURN (ossia praticamente: CHR$(13)).

C - L'ultima fase per la scrittura corretta e' la chiusura del file in uso; la sintassi e':

```
CLOSE 3
```

che informa il COMMODORE 64 che si e' ultimata la fase di registrazione, mentre il num. 3 e' sempre il numero associato al nome del file con l'istruzione OPEN.

Provate per esempio a scrivete i dati anagrafici vostri e dei vostri familiari sul nastro.

Per LEGGERE I DATI dal nastro la procedura e' similare:

A - prima fase: apertura del file in lettura (cioe' in modo 0) valgono le regole gia' descritte, ma il registratore richiedera' la pressione del tasto PLAY.

B - seconda fase: si leggeranno i dati dal nastro con un'istruzione di INPUT# oppure di GET# con la sintassi:

```
INPUT#3,A$
INPUT#3,A
GET#3,A$
```

C - terza fase: come per la scrittura dovrete chiudere il file.

Provate ora a rileggere i dati anagrafici precedentemente memorizzati. Ecco un esempio di scrittura e di esempio di un file:

```
0 REM SCRITTURA DATI
10 OPEN 3,1,2,"DATI ANAGRAFICI"
20 INPUT"NOME';N$:IF N$="" THEN 100
30 INPUT"ETA'";E$
40 INPUT"SESSO";S$
50 INPUT"PROFESSIONE";P$
60 PRINT#3,N$:PRINT#3,E$:PRINT#3,S$:PRINT#3,P$
70 GOTO20
100 PRINT#3,"*":CLOSE3
110 STOP
200 REM LETTURA DATI DOPO AVER RIAVVOLTO IL NASTRO
210 OPEN4,1,0,"DATI ANAGRAFICI"
220 INPUT#4,N$:IF N$="*" THEN 300
230 INPUT#4,E$:INPUT#4,S$:INPUT#4,P$
240 PRINT N$:PRINT E$:PRINT S$:PRINT P$
250 GOTO220
300 CLOSE4:STOP
500 REM MIGLIORATELO PER INTERROMPERE
600 REM LA LETTURA AD OGNI GRUPPO (GET!)
```

USO DELLA STAMPANTE

Di seguito si fa riferimento alla stampante VC-1525 proposta dalla COMMODORE per il VIC ed il COMMODORE 64, ma quanto detto, se non specificato in particolare vale per qualsiasi stampante.
La stampante funzionera' correttamente solo se:

1 - e' stata collegata al COMMODORE 64 col cavo di trasmissione dati (con o senza relativa interfaccia)
2 - se e' stata accesa (per la stampante VC-1525 l'interruttore posteriore non deve essere in modo test e quindi e' su 4 o 5)
3 - se il nastro inchiostrato e' posizionato correttamente

4 - se la carta e' stata posizionata correttamente ed e' libera di muoversi trascinata dai trattori o dal rullo
5 - se il fusibile di protezione stampante non e' bruciato (di solito l'accensione e' segnalata da un LED ossia da un spia luminosa).

Prima di provare a lanciare programmi in Basic o comandi diretti di stampa, anche se vi sembrera' banale, controllate sempre che la stampante sia in grado di poter eseguire i comandi che le verranno passati dal COMMODORE 64.

Poiche' anche la stampante e' una periferica, la procedura di colloquio con il COMMODORE 64 e' la medesima gia' vista per il registratore nel caso di trasferimento dati. Per prima cosa:

```
OPEN 6,4,0
```

Con la OPEN si apre il canale di trasmissione dati num. 6 indirizzato alla periferica num. 4 (controllate che l'interruttore posteriore della stampante VC-1525 sia su 4) con l'indirizzo secondario 0; in particolare:

- il numero dei canali e' un qualsiasi numero compreso tra 1 e 255 (non ne potete aprire ovviamente 255 contemporaneamente, ma un massimo di 10).

- il numero della periferica NP=4 puo' anche essere variato, ad esempio la stampante VC-1525 ha una posizione nell'interruttore sul lato posteriore destro che vi consentira' di avere NP=5; cio' e' comodo nel caso di due stampanti che lavorino contemporaneamente per distinguere le stampe della prima da quelle della seconda

- l'indirizzo secondario (nell'esempio 0) e' un comando che consente di predisporre la stampante per particolari funzioni. Queste variano da stampante a stampante, percio' sara' necessario consultare il manuale specifico; in particolare per la stampante VC-1525 il valore 0 (che puo' anche essere omesso) predispone l'utilizzo del SET caratteri maiuscolo-minuscolo, mentre col valore 7 si passa al SET caratteri grafico-maiuscolo.

Ora per stampare un dato bastera' inviare il comando:

```
PRINT#6,DATO
```

otterrete la stampa del valore della variabile DATO attraverso il canale 6 sulla stampante.

Se le cose da stampare sono tante si puo' anche trasferire l'indirizzo di stampa direttamente alla stampante con l'istruzione:

CMD 6

in questo caso tutti i comandi PRINT (come PRINT DATO oppure LIST, oppure PRINT "PROVA DI STAMPA") vengono eseguiti attraverso il canale 6 sul supporto caratteristico della periferica associata dalla OPEN al canale 6 (cioe' LA CARTA della stampante).
Cio' puo' essere fatto direttamente o da programma: caricate un qualsiasi programma, per esempio il seguente:

```
10 PRINT "PROVA DI ESECUZIONE"
20 PRINT "LISTATO DI PROGRAMMA"
30 PRINT "CON STAMPANTE IN MODO DIRETTO"
```

se battete LIST e lo vedrete ovviamente comparire sul video;

battete invece i seguenti comandi:

OPEN 6,4:CMD 6   <RET>

Vedrete che la stampante dara' un segno della sua esistenza muovendo il carrello: ora il vostro "video" e' il foglio di carta, infatti:

PRINT "HO INTENZIONE DI LISTARE"   <RET>

provochera' la stampa della stringa, mentre:

LIST <RET>

provochera' il listato del programmino.
Fate attenzione che il CMD 6 rimane attivo per tutte le stampe successive fino a che non date l'istruzione PRINT#6. Per capire piu' facilmente questa differenza provate a pensare a due persone che si telefonano:

- l'istruzione OPEN e' l'equivalente della composizione del numero di telefono da parte dell'utente C (COMMODORE 64) per chiamare il sig. S (stampante); se la stampante e' spenta, il telefono continuera' ugualmente a suonare (fino a che verra' posato il ricevitore = istruzione CLOSE).
- quando la stampante e' accesa l'istruzione CMD la pone in ascolto continuativamente, cioe' in attesa, al telefono, dei dati da stampare.

- Sarebbe una scortesia se il sig. C chiudesse il telefono senza avvertire il sig. S che ha finito di parlare con lui, infatti il sig. S resterebbe in ascolto senza capire se il colloquio e' terminato o no:

percio' non usate mai l'istruzione CLOSE 6 per chiudere la comunicazione se stavate stampando con il comando CMD (non si rompe niente ma avrete un cattivo funzionamento anche del video).

Per chiudere correttamente la trasmissione dati e' necessario usare l'istruzione PRINT# che ha la caratteristica di avvertire il sig. S di ascoltare alcuni dati e poi con il telefono aperto continuare pure il suo lavoro (il sig. S potrebbe cosi' dedicarsi all'ascolto di altri telefono collegati con altri canali aperti).

In tal modo ogni volta che l'istruzione PRINT# viene eseguita, la stampante esegue la stampa e successivamente si libera dalla comunicazione fino a che non viene richiamata da un'altra istruzione PRINT# o CMD. In sintesi CON PRINT#6 liberate la stampante dal collegamento, poi chiudete il canale di collegameto con la CLOSE 6, come nei seguenti esempi:

```
OPEN 6,4:CMD6:LIST       <RET>
PRINT#6:CLOSE6           <RET>
```

```
10 OPEN 6,4:PRINT#6,"STAMPA"
20 CMD6:PRINT"SU STAMPANTE"
30 PRINT#6:CLOSE 6
```

Sarebbe errato se la riga 30 fosse solamente:

```
30 CLOSE 6
```

L'istruzione CLOSE e' indispensabile per il buon funzionamento dei vostri programmi (soprattutto se complicati):

- in primo luogo avete a disposizione solo 10 canali contemporaneamente aperti, se dovete lavorare con molti indirizzi secondari e con il drive o col registratore non e' il caso di lasciare aperti dei canali che creino confusione nel programma;

- in secondo luogo una OPEN seguita da una CLOSE appena e' ragionevolmente completata la funzione per cui era stato aperto il canale, rendera' i vostri programmi piu' leggibili anche à distanza di tempo, e quindi piu' facilmente modificabili;

- un errore che vi potra' capitare e' quello di inserire in cicli di FOR NEXT o di IF la OPEN, dimenticandovi la corrispondente CLOSE percui il programma al secondo giro cerchera' di aprire un canale gia' aperto e segnalera' l'errore FILE OPEN ERROR.

- Fate anche attenzione a non chiudere prematuramente il file altrimenti avrete la segnalazione di errore FILE CLOSE ERROR.

- Infine se tentate di aprire piu' di 10 file contemporaneamente avrete segnalazione di errore: ? TOO MANY FILES ERROR IN ...

Ecco un esempio piu' complesso:

```
10 PRINT"STAMPA SU VIDEO"
20 OPEN 9,4:PRINT#9,"SU STAMPANTE":CLOSE 9
30 FOR I=1TO10: PRINT I;:NEXT I
40 OPEN 7,4:CMD7:PRINT"SU STAMPANTE":PRINT
50 FOR I=1TO10: PRINT I;:NEXT I
60 PRINT#7:CLOSE 7
```

Quando si usa l'istruzione CMD la tabulazione avviene su carta nello stesso identico modo che su video, tenendo presente che non avete la possibilita' di far tornare il cursore verso l'alto e che comunque i comandi dei tasti di controllo (CRSV e CRSH) messi tra virgolette non funzionano (cosi' pure il reverse ed il colore!)
Alcune di queste tabulazioni si possono tuttavia ottenere con l'invio di caratteri di controllo sottoforma di:

PRINT CHR$ (num.)

di questi valori alcuni sono standard per tutte le stampanti, mentre altri variano, percio' bisognera' consultare il manuale relativo. Quelli standard sono:

CHR$	RISULTATO
10	il carrello ritorna a capo e salta una riga
13	il carrello ritorna a capo
14	raddoppia la lunghezza dei caratteri (espansi)
15	passa al SET di carattere STANDARD

nel caso della stampante VC-1525 inoltre si possono usare i seguenti caratteri:

8	permette la stampa di un carattere definito in precedenza nel programma
16	usa i due caratteri successivi per tabulatore
17	passa al SET di caratteri minuscolo-maiuscolo
18	accende il modo reverse
145	passa al SET di caratteri grafico-maiuscolo
146	spegne il modo reverse

Il carattere CHR$ (num.) non deve essere necessariamente separato dal resto delle cose da scrivere con segni di interpunzione.
Di seguito viene riportato un esempio:

```
10 OPEN 1,4
20 FOR I= 1 TO 6: READ A(I): NEXT
25 FOR H=1 TO 6
30 PRINT#1,CHR$(A(H));"PROVA CON IL CODICE";A(H);"  ";
40 NEXT:PRINT#1:CLOSE1:END
100 DATA 14,15,18,146,17,145
```

USO ELEMENTARE DELL'UNITA' DISCO

Se l'unita' disco e' stata:
- collegata correttamente,
- e' accesa,
- contiene un dischetto inserito per il verso giusto,

potete cominciare l'interazione tra il COMMODORE 64 e l'unita' disco, ricordandovi inoltre che non e' buona norma lasciare il dischetto nel drive mentre lo state accendendo o spegnendo ed e' altrettanto deleterio rimuovere il dischetto mentre la luce rossa e' accesa per segnalare che il dichetto e' in movimento.
Insieme al drive viene sempre fornito un dischetto chiamato "TEST/DEMO" che contiene dei programmi chiamati UTILITIES necessari per semplificare alcuni comandi e per svolgere determinate funzioni (senza perdere anni-luce per programmare i programmi).
Questo dischetto e' identico agli altri: e' costituito da un supporto in materiale plastico rivestito da una pellicola di ossidi metallici in grado di magnetizzarsi al contatto con la testina di lettura-scrittura dell' unita' disco; il tutto e' protetto da una busta in materiale plastico rigido costruita in modo da non rigare la superficie che deve proteggere. La busta presenta un foro circolare centrale utilizzato dal drive per imprimere la rotazione al dischetto ed una fessura a forma rettangolare e bordi curvilinei per consentire il contatto dischetto testina. Il verso corretto per inserire il disco e' con la fessura rettangolare verso l' alto e verso l' unita' disco.Prima di inserirlo nel drive controllate che tenendo il dischetto in mano come appena detto, la tacca esistente sul lato sinistro sia stata coperta con l'apposito adesivo color argento (chiamato anche WRITE PROTECT) che impedira' al COMMODORE 64 di scrivere sul disco in seguito ad eventuali comandi errati che potreste dare intanto che non siete pratici (E' anche bene che prima di togliere il fatidico adesivo siate riusciti ad ottenere, da chi e' pratico, una copia del suddetto disco).

Sul disco l'insieme di file dati o programmi e' registrato in anelli concentrici detti TRACCE (il loro numero va da 1 a 35 e cresce dall'esterno verso l'interno); ogni traccia e' suddivisa in archi chiamati SETTORI. (il loro numero varia da 20 per le tracce esterne a 16 per quelle interne); in quello centrale si trova l'indice dei programmi residenti sul disco che viene chiamato DIRECTORY.

Questo indice e' un file come tutti gli altri e quindi puo' essere caricato con l'istruzione LOAD scritta come segue:

LOAD "$",8,0   <RET>

La luce rossa si accendera' un istante e successivamente sul video potrete leggere:

SEARCHING FOR $ (fase di ricerca del file $)

LOADING   (caricamento in corso)

READY

ora il directory e' in memoria, lo si puo' vedere con l'istruzione:

LIST.

E' importante ricordare di mettere i caratteri:

,8

dopo il nome del file che volete caricare, infatti avete cosi' specificato il numero della periferica (se lo dimenticate il COMMODORE 64 si disporra' a cercare il programme "$" sul registratore e chiedera' la pressione del tasto PLAY secondo il famoso detto che "chi male intende peggio risponde". Se poi siete felici possessori di piu' unita' dischi potrete, con una utility del TEST/DEMO, modificare tale numero da 8 a 15 in modo da battezzare anche le altre unita' con un numero di riferimento.

Prima di continuare ad esaminare l'istruzione LOAD, visto che sul video compare un directory provate ad osservarlo:

- la prima riga contiene in reverse il nome assegnato al disco nella istruzione di formattazione

- nelle righe seguenti sono segnati come in indice tutti i files registrati su disco secondo l'ordine:

blocchi occupati   nome del file   tipo del file

- infine e' possibile leggere anche il numero di blocchi residui, cioe' non ancora memorizzati e percio' disponibili per la memorizzazione di altri dati (esempio: 432 BLOCKS FREE)

Il secondo numero che avete visto nell'istruzione LOAD (cioe' ,0) e' opzionale, se non viene messo il programma viene caricato all'inizio del Basic, ossia dalla locazione 2048 in avanti, se vale 1 il programma viene caricato in memoria come si trovava al momento della sua memorizzazione su disco.
Oltre a permettervi di caricare il directory, l'istruzione LOAD sara' utile per caricare ache altri programmi (cioe' files programmi) e non sara' nemmeno necessario che sia stato specificato tutto il nome per intero in quanto grazie all'opzione chiamata "PATTERN MATCHING" la parte terminale del nome puo' essere sostituita con il carattere "*" come nell'esempio:

LOAD "PRO*",8

provochera' il caricamento in memoria del primo programma avente iniziali PRO.
Se pero' avete due programmi che iniziano cosi':

PROVA DI STAMPA
PROGRAMMA N. 2

e volete caricare il secondo, l'istruzione corretta sara':

LOAD "PROG*",8

Il comando LOAD "*",8 provoca il caricamento dell'ultimo programma scritto o caricato dal disco, se prima del comando non si e' caricato o memorizzato alcun programma si otterra' allora il caricamento del primo programma presente sull'elenco del directory. Potete anche sostituire alcune lettere a voi non note del nome del programma con il carattere "?", ad esempio:

LOAD "P??G*",8

provochera' la sostituzione di ?? con RO e carichera' il secondo programma. Se tuttavia (ma come siamo pignoli !) esistesse un altro programma con il nome:

PAGG. 26

il comando sarebbe stato ambiguo, perche non specificava

quale dei due:

```
PAGG  26          P??G*
PROGRAMMA N.2     P??G*
```

percio' verra' caricato il primo presente in indice.

FORMATTAZIONE DI UN DISCO

Ora si vedra' come preparare un dischetto per registrare successivamente dei files dati o programmi.
Assicuratevi che il disco da utilizzare NON CONTENGA ALTRI PROGRAMMI (perche' l'operazione successiva li distruggera') e che NON SIA PROTETTO con l'adesivo di cui abbiamo gia' parlato, e date le seguenti istruzioni:

```
OPEN 15,8,15
PRINT#15,"NEW0:NOME,1"
```

- la prima istruzione apre il file logico (o canale) num. 15 alla periferica num. 8 (il disco) con indirizzo secondario 15 che il drive utilizza solo per trasmettere o ricevere istruzioni.
- la seconda provoca la FORMATTAZIONE del disco, ossia la sua suddivisione per settori o tracce per il futuro utilizzo.

NEW serve per indicare che il dischetto deve essere trattato come se fosse nuovo e quindi formattato.

- il numero 0 sulla vostra unita' disco ha poco senso, ma su altre dotate di due dischi serve ad indicare se la formattazione va fatta sul drive 0 o sul drive 1 (il comando purtroppo e' lo stesso per tutte le unita')
- il NOME sara' una stringa alfanumerica assegnata di 16 caratteri mentre il 13 e' un qualsiasi numero da 10 a 99 cioe' con due caratteri: entrambe sono utili per la classificazione del dischetto.

Dopo aver premuto il tasto RETURN potrete udire rumori strani e passera' un po' di tempo durante il quale la luce rossa restera' accesa continuamente. Alla fine, a luce rossa spenta, sarete finalmente in possesso di un dischetto pronto per salvare i vostri programmi; l'istruzione sara':

```
SAVE "NOME PROGRAMMA",8,0    (,0 e' opzionale)
```

dove il NOME PROGRAMMA e' una stringa di lunghezza non superiore ai 16 caratteri, mentre il num. 8 e' il num.

della periferica. Iì num. 0 e' invece un comando che riguarda la locazione di memoria in cui ha inizio il programma, se questo vale 0 il programma sara' memorizzato normalmente, se vale 1 verra' memorizzato con riferimento al preciso punto in cui ha inizio nella memoria del COMMODORE 64.
Il drive prima di registrare sul disco il programma controlla che nel directory non esista un nome uguale a quello assegnato al nuovo file-programma; se cio' accade il comando SAVE non viene eseguito e si avra' il lampeggio del LED rosso del drive per segnalare l'errore.
Per poter scrivere un file con lo stesso nome di uno esistente, perdendo logicamente quello vecchio, dovrete dare questo comando:

```
SAVE "@0:NOME PROGRAMMA",8
```

Ricordatevi di controllare, prima di cominciare a lavorare con un disco, che la memoria disponibile sullo stesso sia sufficiente per poter eseguire successivamente il SAVE del programma. Infatti, se dopo aver dato il comando SAVE lo spazio di memoria risulta insufficiente, avrete segnalazione di errore solamente dal LED del drive che comincera' a lampeggiare, poiche' questa non e' una segnalazione molto efficace, potreste correre il rischio di perdere il programma con la convinzione di averlo salvato correttamente.

I comandi esposti sono semplificabili caricando dal dischetto del TEST/DEMO il programma chiamato C-64 WEDGE e dando RUN, da questo momento in poi:

/ sostituisce il LOAD senza bisogno delle virgolette
↑ sostituisce il LOAD ed il RUN contemporaneamente
@ o > danno comandi al disco senza bisogno di eseguire: OPEN 15,8,15 e PRINT#15
@ o > seguito da $ caricano e listano il directory senza alterare il programma in memoria (!!)
> da solo permette di visualizzare il tipo di errore che ha provocato il lampeggio del drive.

L'unico inconveniente e' che questi comandi cosi' semplici sono solo di uso immediato, cioe' non possono essere inseriti in programmi Basic e vengono persi con lo spegnimento del COMMODORE 64

Usando i comandi visti provate a registrare un file di dati sequenziali.

```
0 GOTO 1000
5 REM SCRITTURA DATI
10 OPEN 3,8,3,"DATI ANAGRAFICI,S,W"
15 PRINT"* PER FINIRE":PRINT
20 INPUT"NOME";N$:IF N$="*" THEN 100
30 INPUT"ETA'";E$
40 INPUT"SESSO";S$
50 INPUT"PROFESSIONE";P$:PRINT:PRINT
60 PRINT#3,N$:PRINT#3,E$:PRINT#3,S$:PRINT#3,P$
70 GOTO20
100 PRINT#3,"*":CLOSE3:RETURN
110 REM *********************
200 REM LETTURA DATI
210 OPEN4,8,4,"DATI ANAGRAFICI,S,R"
220 INPUT#4,N$:IF N$="*" THEN 300
230 INPUT#4,E$:INPUT#4,S$:INPUT#4,P$
240 PRINT"NOME",N$
241 PRINT"ETA'",E$
242 PRINT"SESSO",S$
243 PRINT"PROFESS.",P$:Q$=""
245 PRINT:PRINT:PRINT"BATTERE UN TASTO":PRINT:PRINT
250 GETQ$: IF Q$="" THEN 250
260 GOTO220
300 CLOSE4:RETURN
400 END
1000 PRINT:PRINT:PRINT"SCELTA"
1010 PRINT:PRINT"1 FORMAZIONE FILE DATI"
1020 PRINT:PRINT"2 LETTURA FILE DATI"
1025 PRINT:PRINT'3 FINE":PRINT
1030 INPUT A:PRINT:PRINT
1040 ON A GOSUB 10,200,400
1050 GOTO 1000
```

Avrete notato che dopo il nome del file compaiono due sigle:

S serve per specificare che tipo di file intendiamo aprire, le altre possibili sigle sono:

P = PROGRAMMA
R = FILE RELATIVO
U = FILE DI COMANDI

la seconda sigla significa il tipo di apertura del FILE

R = READ IL FILE E' APERTO IN LETTURA
W = WRITE E' APERTO IN SCRITTURA

Ricordatevi che la mancanza dell'istruzione CLOSE NC alla fine delle operazioni di scrittura dei dati su disco provoca la perdita del FILE stesso, infatti sul directory comparira' un asterisco in parte alla sigla che specifica il tipo di e quest'ultimo sara' caratterizzato da un'occupazione di 0 blocchi.

Tutte le volte che avete salvato un programma su disco (come del resto su cassetta) e' buona norma eseguire l' istruzione:

VERIFY "NOME",8

che vi permettera' di verificare se la memorizzazione del programma e' stata effettuata correttamente; poiche' richiederete la verifica di un programma appena registrato, sara' inutile ripetere il nome che potra' essere semplicemente sostituito col carattere *

Il COMMODORE 64 vi rispondera':

```
SEARCHING FOR NOME    ( ricerca il file da verificare)
VERIFYNG              ( verifica in corso )
OK                    ( la verifica e' positiva )
```

READY

viceversa dovrete rieseguire il SAVE del programma.

Per esaminare altri particolari sui comandi OPEN, PRINT etc, consultate anche la descrizione dei comandi Basic nel Cap 4 del libro.

CAPITOLO 7

PROGRAMMI DI USO GENERALE

COSA SONO E A COSA SERVONO

Le Ditte produttrici di programmi, che prima scrivevano quasi esclusivamente software su misura per applicazioni specifiche, oggi, grazie alla diffusione dei personal e home computers, hanno sviluppato programmi di uso generico.
Tali programmi si adattano, nell'ambito di una certa problematica, alle esigenze dell'utilizzatore, senza bisogno di modifiche di sorta.
Ad esempio, il primo dei programmi di cui parliamo di seguito, il DATA BASE, permette di creare archivi di dati di qualsiasi genere, a scelta dell'utente, contenenti informazioni di qualsiasi tipo e permette all'utente di ricercare le informazioni archiviate con criteri di scelta pure essi definibili di volta in volta.
In pratica permette all'utente di programmare la propria gestione di blocchi di informazioni o di dati, senza che egli debba scrivere una sola riga di programma per fare cio'.
Lo stesso accade col terzo dei programmi descritti di seguito, il FOGLIO ELETTRONICO, il quale permette invece di programmare calcoli di qualsiasi genere, a misura dei problemi piu' disparati che l'utente intende risolvere, senza bisogno che egli conosca, anche in questo caso, una sola istruzione per la programmazione.
Ma andiamo con ordine.
I programmi di uso generico di cui parleremo nei prossimi paragrafi sono:

1 - DATA BASE
2 - WORD PROCESSOR
3 - FOGLIO ELETTRONICO

DATA BASE
(ovvero l'organizzazione dei dati)

Per capire che cosa vuol dire creare una organizzazione dei dati partiamo dal concetto di organizzazione delle cose in senso figurato.
Noi normalmente organizziamo le cose in funzione di quello che desideriamo poi ottenere dalle stesse, oppure di che cosa intendiamo farci.
Facciamo l'esempio dei piatti.
Se in un ristorante, o anche a casa nostra se abbiamo piu'piatti di quanti effettivamente siamo (a me non accade mai perche' li rompo le rare volte che li lavo), mettiamo tutti piatti cosi' come vengono uno sull'altro, avremo dei problemi al momento di tirare fuori un fondo, un piano ed uno da frutta : dovremo infatti cercare nella pila di piatti quello che ci serve spostando tutti quelli che ci stanno sopra prima di poterlo prendere.
Per organizzarci abbiamo di certo piu' possibilita': tutto sta ad avere le idee chiare su dove ed a che costo si intende raggiungere il risultato atteso, che in questo caso particolare e' quello di poter prelevare nel piu' breve tempo possibile e con il minimo sforzo i piatti necessari alla pappa.

1 - possiamo fare tante pile di piatti ciascuna per ciascun tipo
2 - possiamo fare una pila di piatti in sequenza per coperto (un fondo, un piano, un frutta ... un fondo, un piano, un frutta ... etc) .
3 - possiamo fare una sola pila con tutti i piatti di un tipo insieme e poi con tutti quelli dell'altro tipo insieme e cosi'via.
4 - possiamo metterli in pila tutti a casaccio come abbiamo detto poco sopra.
5 - possiamo metterli ordinati per tipo su di una apposita rastrelliera (tipo scolapiatti, tanto per intenderci) .
6 - possiamo metterli a casaccio su di una apposita rastrelliera
7 - e chi piu' ne ha piu' ne metta ....

Tutto il problema sta nel decidere quale di queste organizzazioni e' la migliore ; i criteri sono sia funzionali che oggettivi e perche' no, anche soggettivi.

La prima soluzione ci permette di prelevare velocemente ed anche di depositare velocemente i piatti, ma ci impone di mettere le pile grandi o piccole che siano in posti diversi e di dedicare loro un certo spazio fisso

che puo' essere magari troppo rispetto al volume realmente occupato

La seconda soluzione ci permette di prelevare velocemente ma al momento di depositare ci dobbiamo organizzare in modo da preparare i piatti ordinati in serie per coperto prima di metterli in pila .

La terza soluzione ci permette di avere le ideee chiare su quanti piatti di ciascun tipo abbiamo, ma non e' eccessivamente pratica ne' per prelevare ne' per riporre, a meno di non sapere esattamente di quanti piatti si ha bisogno e li si preleva in una volta sola sollevando quelli che non servono per un massimo di due operazioni di sollevamento, quindi, per ogni prelievo ; anche in questo caso il deposito presuppone di organizzare i piatti per tipo prima di riporli.

La quarta soluzione ci fa capire che abbiamo un sacco di piatti ma che la loro gestione e' alquanto complessa e che prima o poi finiremo per combinare un disastro facendo cadere tutto .

La quinta soluzione permette una rapida ed immediata gestione sia del prelievo che del deposito, ma presuppone l'organizzazione della rastrelliera ed il lasciare sempre tanti buchi vuoti quanti sono i piatti, con tanto spreco quanto si eccede nel lasciare lo spazio.

La sesta soluzione permette una rapida gestione del prelievo, ed una ancor piu' immediata gestione del deposito, senza probabilmente grossi sprechi in quanto faremo in modo di avere una rastrelliera ad elementi componibili.

MA CHE COSA DIAVOLO C'ENTRA QUESTO CON UN DATA BASE ???

C'entra eccome !!!

Immaginiamo infatti che i nostri piatti siano i dati e che i sei modi proposti siano sei modi con cui un programma puo' cercare o scrivere i dati che noi vogliamo sui nostri dischetti .
L'architettura dei nostri archivi dipendera' quindi da QUALE programma avremo scelto e la velocita', l'occupazione di memoria, la occupazione di spazio sui supporti magnetici, ... tutto dipendera' dal tipo di DATA BASE che avrete adottato, in relazione a che tipo di applicazione intendete svolgere.

UN DATA BASE E' QUINDI UN MODO DI ORGANIZZARE I PIATTI, pardon, I DATI IN FUNZIONE DI COME SI PENSA DI ACCEDERVI SIA IN LETTURA CHE IN AGGIORNAMENTO, ED IN FUNZIONE DELLA FREQUENZA E DEI MODI DI TALI AZIONI .

Esistono quindi organizzazioni di tipo sequenziale, di tipo a chiave sequenziale, di tipo indicizzato, di tipo gerarchico e di tipo relazionale , ... e chi piu' ne ha piu' ne metta ......

Un'organizzazione di tipo sequenziale presuppone, ad esempio, che per ricercare un dato, si parta dall'inizio dell'archivio e si leggano tutti i dati presenti confrontandoli con il cercato ...(ed e' l'unico gestibile con il registratore a cassette).

Un'organizzazione di tipo indicizzato e' fatta in modo che mediante una "chiave di ricerca" appositamente strutturata fin dall'inizio , sia possibile accedere direttamente al dato cercato mediante la sua "chiave" .

Un data base di tipo relazionale permette di ricercare liberamente i dati e addirittura le loro connessioni con altri dati senza quasi che sia stato necessario predefinirne "chiavi" o "relazioni".

IN PRATICA QUALSIASI ARCHIVIO DI DATI E' UN DATA BASE, ORGANIZZATO IN FORME ED IN MODI E CON POTENZE DIVERSE.

Ma vediamo ora quali applicazioni tipiche si possono realizzare su di un personal computer utilizzando uno di questi data base evoluti.

I campi sono vastissimi.
Ho recentemente realizzato un data base per un mio amico che voleva tenere nota di tutte le telefonate che riceveva nel suo studio tramite la sua segretaria e poi poter ricercare qualsiasi parola fosse stata annotata (dal nome, ad una delle parole argomento della telefonata, alla data , all'ora ....) senza far uso di chiavi predefinite o cose del genere.
Un altro mio amico medico voleva avere uno schedario per ogni suo paziente con vita morte (speriamo di no se no perde il cliente) e miracoli di ciascuno di essi, incluse informazioni "non formattate", cioe' a chiave libera , quali l'annotazione di fatti e argomenti non tipicamente "medici": ad esempio, voleva poter annotare il fatto che la signora Tal Dei Tali si era comprata un cagnolino, cosi' da ricordarsi di chiedere notizie in occasione della visita successiva (... ed ecco la politica nella medicina ...)

Posso altresi' decidere di archiviare il mio elenco dei dischi e delle cassette (di MUSICA, non di dati e programmi) , in modo che posso ricercare che numero di disco e' la quinta sinfonia di Beethoven eseguita da Bernstein tra le otto o nove quinte sinfonie che per fortuna posseggo, o la cassetta di Flashdance da sentire con la fanciulla a cui piace tanto il tema d'amore ......

Bisogna pero' rimettere i dischi in ordine di numero se no non serve a niente !

Oppure posso creare l'archivio delle "COSE DA FARE" con tanto di "richiesto da" e per quando e che cosa ci sto facendo ed a che punto sono e quindi lo scadenziario automatico di quello che devo e posso fare.

Oppure le ricette di cucina con tanto di lista della spesa automatica se gestisco anche il contenuto del frigo e della dispensa ....

Oppure la rubrica dei numeri telefonici

Oppure la agenda degli appuntamenti

Oppure la raccolta delle spese fatte a fronte di un certo budget

Oppure ...... QUELLO CHE ORA VI PASSA PER LA MENTE, CHE VI POSSA FAR COMODO POTER G E S T I R E , TERMINE CHE ALTRO NON VUOL DIRE CHE POTER RICERCARE ED AGGIORNARE CON LA MASSIMA RAPIDITA' .

E per finire un piccolo grande segreto.

Prima di fare qualsiasi passo, sedetevi a pensare che cosa volete, come e quando ed a che costo, ma soprattutto CHE COSA e PER COSA . In Informatica le varianti in corso d'opera, i "colpi di timone", sono quanto di peggio possa capitare, e questo capita sicuramente a chi non sa cosa vuole (meditate gente...)

IL TRATTAMENTO DELLA PAROLA (WORD PROCESSING)

Una ulteriore grossa area applicativa per l'utilizzo di un personal computer nella nostra vita sia professionale che privata e' quella del trattamento della parola (scritta) detto anche, anzi, soprattutto WORD PROCESSING

Mi ricordo che ai suoi tempi fece notizia il Presidente Nixon per il fatto che inizio' a scrivere le sue memorie utilizzando un personal computer (ed allora ce n'erano ben pochi in giro).
La peculiarita' principale di un sistema di elaborazione testi e' che permette di scrivere di getto, quasi alla rinfusa, cio' che viene in mente , con la possibilita' di modificarlo in ogni sua parte senza dover riscrivere tutto da capo al momento di battere la bella copia .
Un sistema di word processing gestisce infatti automaticamente l'allineamento delle parole all'interno delle righe e dei margini, che possono essere dinamicamente variati senza dover riscrivere tutto da capo .
Gestisce addirittura nelle sue versioni piu' evolute gli "a capo", con lo spezzamento della parola, utilizzando un dizionario che conosce tutte le possibilita' ammesse dal lessico e che puo' a volte addirittura verificare ed autocorreggere eventuali errori di ortografia avvenuti in fase di battitura del testo .
Il Word Processing da' la possibilita' di scrivere i titoli in grassetto e le formule in compresso con soprascritti e sottoscritti , ottenendo stampe simili ad un libro.
Gestisce la combinazione tra vari testi, permettendo di andare a prelevare pezzi di testo o interi testi da altre parti e portarseli all'interno di quanto si sta scrivendo, evitando perdite di tempo per la scrittura di cose gia' stese ed archiviate in precedenza.
Da' anche la possibilita' di combinare i testi con un indirizzario, in modo da far comporre al calcolatore lettere di tipo "personalizzato" e costruire una emissione di lettere in funzione di una certa necessita' (ad esempio potete decidere di inviare una lettera a tutti quelli che si chiamano PIPPO)

Ma la caratteristica fondamentale di un sistema di trattamento testi e' proprio quella di permettere la correzione dinamica di cio' che si e' scritto direttamente a video, con possibilita' di lasciare al calcolatore ed alla sua stampante l'onere di riscrivere

tutto da capo ... E SENZA FARE ERRORI !!!, nel numero di copie desiderato.

Ancora due parole sulle periferiche che interessano il trattamento dei testi ; a parte il monitor video, ci vuole una stampante su cui ottenere il risultato sia intermedio che finale in stampa su carta . Tali stampanti possono essere di vario livello in quanto a qualita' di stampa e velocita'. In altra sede all'interno di codesto tomo trattasi anche di cio'.

A proposito : questo libro e' stato completamente scritto con un sistema di trattamento testi, e neanche dei piu' complessi ed evoluti !

IL FOGLIO DI CALCOLO ELETTRONICO (WORKSHEET)

Una delle applicazioni che ha avuto maggior fortuna e diffusione al mondo nel settore degli home e personal computer e' sicuramente il trattamento automatico del calcolo matriciale : il cosiddetto foglio di calcolo elettronico o worksheet.
L'esempio piu' lampante e' il numero di copie vendute in tutto il mondo del famosissimo programma VISICALC (VISICALC e' um marchio registrato della Visicorp USA), il piu' diffuso ed imitato in questo genere di programmi .

L'utilizzo pratico di un sistema di calcolo matriciale elettronico e' molto ampio : va dalle simulazioni di vendita in funzione dei livelli di produzione di un'azienda , alla gestione del budget personale o familiare.

Ma vediamo brevemente come funziona.
Si tratta di un insieme di righe e colonne referenziate da un codice (ad esempio lettere per le colonne e numeri per le righe, proprio come in una battaglia navale). Ciascuna coordinata puo' contenere un dato numerico, una formula che collega ed elabora algebricamente o logicamente piu' dati, o un insieme di lettere qualsiasi quali quelle componenti un titolo .
In pratica se mettiamo nella coordinata A1 (Colonna A, Riga 1) il valore 2 e nella coordinata B1 (Colonna B, Riga 1) il valore 2, e vogliamo avere nella coordinata C1 (Colonna C, Riga 1) il risultato della somma di A1+B1, in tale coordinata scriveremo A1+B1 , e il nostro foglio di lavoro elettronico ci fara' vedere come risultato 4 .

Ma il bello non e' finito, anzi, arriva adesso !
Infatti se adesso vogliamo cambiare il contenuto della coordinata B1 da 2 a 5, ad esempio , scriveremo 5 in B1, e il sistema calcolera' automaticamente in C1 2+5, e cioe' 7.
Cio' e' molto utile per tutti quei lavori di calcolo ripetitivo sotto varie condizioni, o per effettuare simulazioni in modo da vedere che cosa succede se si varia un parametro e che effetti ha sul risultato finale o sulle singole componenti dello stesso .

Personalmente utilizzo tale programma per un gran numero di applicazioni e non solo sul mio personal, ma per fortuna anche sul grosso elaboratore di cui dispone la mia azienda ove per la prima volta proprio recentemente e' stato proposto e subito installato dato la sua incredibile versatilita' ed indubbia comodita' applicativa un programma di questo tipo.

Spiegare a parole come funzionano questi programmi e' abbastanza difficile. Noi ci abbiamo provato, ma crediamo che vi convenga provarli personalmente per capirne appieno la potenza e versatilita'.
Anche questo, come gli altri programmi descritti, e' di utilizzo generico ed e' quindi, come tale, adattabile all'uso specifico che l'utente ne vuol fare, per risolvere qualsivoglia problema in tema con l'argomento del programma stesso.

CAPITOLO 8

TAVOLE

TABELLA RIASSUNTIVA ALFABETICA DEI COMANDI BASIC

Questo elenco in ordine alfabetico vi servira' per una rapida consultazione ogniqualvolta abbiate un dubbio o vogliate vedere l'esatta sintassi di un comando .

ABS( )	Da' il valore assoluto di un numero o di una variabile.(il valore senza il segno) es. PRINT ABS(-3.14159) Da' 3.14159
AND	Operatore logico usato anche nell' istruzione IF..THEN. es. IF A = 1 AND B = 1 THEN GOTO 100
ASC( )	Da' il numero corrispondente della tabella ASCII del carattere in argomento. es. PRINT ASC("A") Da' 65 (vedi tabella)
ATN( )	Da l'arcotangente del numero in argomento. es. PRINT ATN(12) Da' 1.4876551
CHR$( )	Da' il carattere corrispondente al numero in argomento secondo il codice ASCII. es.PRINT CHR$(65) Da' A
CLOSE	Chiude il canale della periferica o il file. es.CLOSE 4 chiude il canale 4
CLR	Mette tutte le variabili a zero. es. 10 CLR
CMD	Mette in uscita il file o la periferica aperta sullo stesso canale. es. 10 OPEN 1,4 20 CMD 1 30 LIST Da' il LIST del programma sulla periferica 4

CONT	Fa proseguire l'esecuzione di un programma interrotto da un'istruzione di STOP o END .
es. CONT

COS ( )	Da' il Coseno del numero o variabile in argomento.
es. PRINT COS(12)   Da' .84385396

DATA	Serve per inserire stringhe o numeri che verranno letti dall'istruzione READ.
es. 100 DATA 100,PIPPO,2000,"RAOUL"

DEF FN()	permette di definire una funzione richiamabile da programma.
es.10 DEF FNE(A) = A*A
   20 PRINT FNE(2)   Da' 4

DIM	Dimensiona il n massimo di elementi per il vettore della variabile in argomento.
es.10 DIM A(100)

END	Fa terminare l'esecuzione del programma .
es. 1000 END

EXP ()	Da' e=2.718289 elevato al valore dell'argomento
es. PRINT EXP(9) Da' 8103.08393

FOR	Parte del ciclo FOR/NEXT che specifica il valore dell 'inizio e della fine del ciclo.
es. 10 FOR I = 1 TO 100

FRE ()	Da' il numero di bytes liberi nella memoria.
es.PRINT FRE(0)

GET	Legge un carattere dalla tastiera e lo mette nella variabile specificata.
es.GET B$

GET#	Legge un carattere dalla periferica o file precedentemente aperti.
es. GET # 2,A$

GOSUB	Salta al sottoprogramma che sta al numero di linea indicata.
es. GOSUB 1000

GOTO	Salta incondizionatamente al numero di linea specificato.
es. GOTO 100

IF/THEN — Istruzione decisionale. Esegue quanto specificato dopo il THEN se la condizione dopo IF e' soddisfatta.
es. IF A=2 THEN GOTO 100

INPUT — Arresta l'esecuzione di un programma e attende che vengano battuti dei dati da tastiera che verranno messi nella variabile specificata.
es. 10 INPUT A$

INPUT# — Prende un dato dalla periferica o file precedentemente aperto.
200 INPUT# 1,A$

INT() — Da' l'intero del numero o variabile in argomento.
es.PRINT INT(12.3) Da' 12

LEFT$ — Ritorna il numero di caratteri specificati in argomento a sinistra.

```
es. 10 A$="GIOVANNI"
    20 PRINT LEFT$ (A$,4)   Da' ANNI
```

LEN() — Ritorna la lunghezza della stringa in argomento
es.A$="LUNGO":PRINT LEN(A$)   Da' 5

LIST — Da' la lista delle istruzioni del programma in memoria.
es. LIST

LOAD — Legge un programma dalla periferica specificata.
es. LOAD"PIPPO",1

LOG() — Da' il logaritmo naturale della variabile o numero in argomento.
es. PRINT LOG(12)   Da' 2.48490665

MID$ — Da' la parte di stringa specificata in argomento.
es. PRINT MID$ ("SCOLLEGARE",2,7) Da' COLLEGA

NEW — Cancella il programma dalla memoria.
es. NEW

NEXT — E' l'istruzione che chiude il ciclo FOR NEXT.

```
es. 10 FOR I = 1 TO 100
    20 PRINT I
    30 NEXT
```

NOT Negazione logica.
es. IF NOT B=1 THEN 100

ON..GOTO Serve per definire un salto incondizionato alla serie di numeri di linea che segue, in base al valore della variabile dopo ON.
es.ON A GOTO 100,200,300

ON..GOSUB Serve per comandare un salto condizionato alla serie di numeri di linea specificati dopo il GOSUB ,in base al valore della variabile specificata dopo ON.
es.ON A GOSUB 100,200,300

OPEN Serve per aprire il colloquio con una periferica o ad aprire un file.
es. OPEN 1,1,1,"NOME"
OPEN 2,4

OR Operatore logico .
es. IF A=1 OR B=1 THEN GOTO 1000

PEEK Legge il valore in decimale contenuto nella locazione di memoria specificata.
es. A=PEEK(27732)

POKE Scrive nella locazione di memoria desiderata il valore specificato in argomento .
es. POKE 650,128

POS() Serve per avere la corrente posizione del cursore sulla riga.
es. 10 A=POS(0)

PRINT Serve per scrivere il valore di una variabile o stringa. Si puo' abbreviare con ?.
es. PRINT "BABBO ";1;A;A$+B$

PRINT# Manda in uscita sulla periferica o file specificato dalla OPEN.
es. 10 OPEN 1,1,1,"PIPPO"
20 PRINT#1,A$;" SOLO "

READ Serve per leggere i valori specificati dall'istruzione DATA e metterli in una variabile.
es. READ A$,A,B

REM Istruzione che viene ignorata. Si usa solo per inserire utili ed indispensabili commenti all' interno di un programma.
es. GOTO 100 :REM SALTA ALLA FASE XY

RESTORE Fa in modo che la prossima istruzione di READ ricominci a leggere dal primo dato contenuto nella serie dei DATA.
es. 100 RESTORE

RETURN Ritorna l'esecuzione del programma alla istruzione successiva al comando GOSUB che ha provocato il salto alla subroutine.
es.1230 RETURN

RIGHT$ Restituisce la parte destra della stringa specificata, per n caratteri, quanti specificati in argomento.
es. 10 A$="LANCIAMISSILE"
    20 PRINT RIGHT$(A$,7) Da' MISSILE

RND () Genera un numero casuale minore di 1 e maggiore di 0.
es. 10 A= RND(0)

RUN Esegue il programma in memoria.
es. RUN

SAVE Memorizza il programma in memoria sulla cassetta o sul disco.
es. SAVE " NOME ",1,1
    SAVE " NOME ",8,1

SIN () Da' il seno del numero o variabile in argomento
es. PRINT SIN(12)   Da' -.536572917

SPC () Stampa il numero di spazi specificati in argomento.
es. PRINT "PIPPO";SPC(3);"PLUTO"
    Da' PIPPO   PLUTO

SQR () Da' la radice quadrata del numero o variabile in argomento.
es. PRINT SQR(9)   Da' 3

STOP Ferma l'esecuzione del programma e stampa il numero di linea dove si trova l'istruzione STOP. Si puo' riprendere l'esecuzione del programma usando CONT.
es. 190 STOP

STR$() Mette il numero o la variabile numerica in argomento in una stringa.
es. 10 A=12
    20 A$=STR$(A)

SYS Chiama ed esegue una routine in linguaggio macchina che risiede alla locazione di memoria specificata dopo la SYS.
es. SYS 58692

TAB() Posiziona il cursore alla posizione specificata in argomento.
es PRINT TAB(12);"SONO ALLA 12 POSIZIONE"

TAN () Da' la tangente del numero o variabile in argomento, espresso in radianti.
es. PRINT TAN(12) Da' -.635859926

VAL () Mette in una variabile numerica il contenuto numerico di una stringa.
es.
```
10 A$="12 PAPPAGALLI"
20 A=VAL(A$)
30 PRINT A          Da' 12
```

TABELLA RIASSUNTIVA ABBREVIAZIONI DEI COMANDI BASIC

COM	ABBR	VIDEO	COM	ABBR	VIDEO
ABS	A SHIFT B	A	NOT	N SHIFT O	N
AND	A SHIFT N	A	ON	NON C' E'	ON
ASC	A SHIFT S	A	OPEN	O SHIFT P	O
ATN	A SHIFT T	A	OR	NON C' E'	OR
CHR$	C SHIFT H	C	PEEK	P SHIFT E	P
CLOSE	CL SHIFT O	CL	POKE	P SHIFT O	P
CLR	C SHIFT L	C	POS	NON C' E'	POS
CMD	C SHIFT M	C	PRINT	?	?
CONT	C SHIFT O	C	PRINT#	P SHIFT R	P
COS	NON C' E'	COS	READ	R SHIFT E	R
DATA	D SHIFT A	D	REM	NON C' E'	REM
DEF	D SHIFT E	D	RESTORE	RE SHIFT S	RE
DIM	D SHIFT I	D	RETURN	RE SHIFT T	RE
END	E SHIFT N	E	RIGHT$	R SHIFT I	R
EXP	E SHIFT X	E	RND	R SHIFT N	R
FN	NON C' E'	FM	RUN	R SHIFT U	R
FOR	F SHIFT O	F	SAVE	S SHIFT A	S
FREE	F SHIFT R	F	SGN	S SHIFT G	S
GET	G SHIFT E	G	SIN	S SHIFT I	S
GET#	NON C' E'	GET#	SPC(	S SHIFT P	S
GOSUB	GO SHIFT S	GO	SQR	S SHIFT Q	S
GOTO	G SHIFT O	G	STATUS	ST	ST
IF	NON C' E'	IF	STEP	RT SHIFT E	ST
INPUT	NON C' E'	INPUT	STOP	S SHIFT T	S
INPUT#	I SHIFT N	I	STR$	ST SHIFT R	ST
INT	NON C' E'	INT	SYS	S SHIFT Y	S
LEFT$	LE SHIFT F	LE	TAB(	T SHIFT A	T
LEN	NON C' E'	LEN	TAN	NON C' E'	TAN
LET	L SHIFT E	L	THEN	T SHIFT H	T
LIST	L SHIFT I	L	TIME	TI	TI
LOAD	L SHIFT O	L	TIME$	TI$	TI$
LOG	NON C' E'	LOG	USR	U SHIFT S	U
MID$	M SHIFT I	M	VAL	V SHIFT A	V
NEW	NON C' E'	NEW	VERIFY	V SHIFT E	V
NEXT	N SHIFT E	N	WAIT	W SHIFT A	W

Nella prima colonna contrassegnata dalla dicitura COM sono riportati i comandi Basic per esteso.
Nella seconda colonna contrassegnata dalla dicitura ABBR.: le abbrevviazioni dei relativi comandi.
Nella terza colonna contrassegnata dalla dicitura VIDEO: i simboli che appaiono sul video usando le abbrevviazioni

TABELLA DEI CODICI ASCII - DECIMALI -

La prima e la seconda colonna rappresentano i simboli che potete ottenere dalle istruzioni POKE e CHR$ inserendovi i corrispondenti valori decimali

I SET	II SET	POKE	CHR$	I SET	II SET	POKE	CHR$
@	@	0	64	SPAZIO		32	32
A	a	1	65	!	!	33	33
B	b	2	66	"	"	34	34
C	c	3	67	#	#	35	35
D	d	4	68	$	$	36	36
E	e	5	69	%	%	37	37
F	f	6	70	&	&	38	38
G	g	7	71	'	'	39	39
H	h	8	72	(	(	40	40
I	i	9	73	)	)	41	41
J	j	10	74	*	*	42	42
K	k	11	75	+	+	43	43
L	l	12	76	,	,	44	44
M	m	13	77	-	-	45	45
N	n	14	78	.	.	46	46
O	o	15	79	/	/	47	47
P	p	16	80	0	0	48	48
Q	q	17	81	1	1	49	49
R	r	18	82	2	2	50	50
S	s	19	83	3	3	51	51
T	t	20	84	4	4	52	52
U	u	21	85	5	5	53	53
V	v	22	86	6	6	54	54
W	w	23	87	7	7	55	55
X	x	24	88	8	8	56	56
Y	y	25	89	9	9	57	57
Z	z	26	90	:	:	58	58
[	[	27	91	;	;	59	59
£	£	28	92	<	<	60	60
]	]	29	93	=	=	61	61
↑	↑	30	94	>	>	62	62
←	←	31	95	?	?	63	63
				@	@	64	64

TABELLA DEI CODICI ASCII -DECIMALI

Si richiama l' attenzione sul fatto che alcuni simboli saranno differenti in funzione del modo carattere (minuscolo-maiuscolo oppure maiuscolo-grafico) in cui si trova il COMMODORE 64 in quell' istante.

I SET	II SET	POKE	CHR$	I SET	II SET	POKE	CHR$
♠	A	65	97	SPAZIO		96	160
│	B	66	98	▌	▌	97	161
─	C	67	99	▄	▄	98	162
─	D	68	100	▔	▔	99	163
─	E	69	101	▁	▁	100	164
─	F	70	102	▏	▏	101	165
│	G	71	103	▒	▒	102	166
│	H	72	104	▕	▕	103	167
╮	I	73	105	▒	▒	104	168
╰	J	74	106	◤	▒	105	169
╯	K	75	107	▕	▕	106	170
L	L	76	108	├	├	107	171
╲	M	77	109	▗	▗	108	172
╱	N	78	110	└	└	109	173
┌	O	79	111	┐	┐	110	174
┐	P	80	112	▂	▂	111	175
●	Q	81	113	┌	┌	112	176
─	R	82	114	┴	┴	113	177
♥	S	83	115	┬	┬	114	178
│	T	84	116	┤	┤	115	179
╭	U	85	117	▎	▎	116	180
╳	V	86	118	▍	▍	117	181
○	W	87	119	▐	▐	118	182
♣	X	88	120	▀	▀	119	183
│	Y	89	.121	▃	▃	120	184
♦	Z	90	122	▅	▅	121	185
+	+	91	123	┘	✓	122	186
▒	▒	92	124	▖	▖	123	187
│	│	93	125	▝	▝	124	188
π	▚	94	126	┘	┘	125	189
◥	▒	95	127	▘	▘	126	190
				▚	▚	127	191

MESSAGGI DI ERRORE.

BAD DATA ERROR	Tentativo di leggere da un file dei dati alfanumerici usando una variabile numerica.
BAD SUBSCRIPT ERROR	E' stato usato un elemento in piu' di quel vettore di quanto dichiarato nel dimensionamento con l'istruzione DIM.
CAN'T CONTINUE ERROR	Dopo un'istruzione di CONT non possibile ,perche' si e' verifi cato un errore o perche' e' sta ta modificata un'istruzione.
DEVICE NOT PRESENT ERR	La periferica chiamata non esi- ste o non e' collegata.
DIVISION BY ZERO ERROR	Tentativo di eseguire una divi- sione per zero.
EXTRA IGNORED	Sono state date piu' risposte di quante richieste da INPUT.
FILE NOT FOUND	Non esiste nessun FILE con il nome desiderato.
FILE NOT OPEN ERROR	Tentativo di lavorare su di un FILE non ancora aperto.
FORMULA TOO COMPLEX ERR	Troppe parentesi in una formula o stringa troppo lunga da trat- tare.
ILLEGAL DIRECT ERROR	Utilizzo di una istruzione che non permette l'esecuzione imme- diata.
ILLEGAL QUANTITY ERROR	Valore usato come argomento in una funzione e' di valore fuori dal consentito.
LOAD ERROR	Durante il caricamento di un programma da nastro e' stata riscontrata una anomalia.
NEXT WITHOUT FOR ERROR	E' stata trovata un' istruzione NEXT senza aver trovato prima il relativo FOR.

NOT INPUT FILE ERROR	Tentativo di leggere un FILE aperto in modo solo scrittura.
NOT OUTPUT FILE ERROR	Tentativo di scrivere in un FILE aperto in modo solo lettura
OUT OF DATA ERROR	Esecuzione di un' istruzione READ senza avere usato DATA, o tentativo di leggere altri DATA non esistenti.
OUT OF MEMORY ERROR	Non vi e' piu' memoria libera per il programma o per le varia bili. Troppi FOR/NEXT o troppi GOSUB RETURN.
OVERFLOW ERROR	Il risultato di un' operazione o l' assegnazione ad una varia- bile e' fuori dal limite consen tito.
REDIM'D ARRAY ERROR	Tentativo di dimensionamento di un vettore o matrice per la se- conda volta durante l'esecuzio- ne del programma.
REDO FROM START	E' stato battuto un carattere alfabetico quando l'INPUT aspet tava un numerico.
RETURN WITHOUT GOSUB ERR	E' stata trovata un' istruzione RETURN senza il relativo GOSUB.
STRING TOO LONG ERROR	Tentativo di [illegible] una stringa piu' lunga [illegible] caratteri.
SINTAX ERROR	Errore di sintassi,e' probabile che sia stata scritta in manie- ra inesatta un'istruzione BASIC
TYPE MISMATCH ERROR	Uso di una stringa al posto di un numero o viceversa.
UNDEF'D STATEMENT ERROR	Chiamata ad un numero di linea che non esiste.
VERIFY ERROR	Durante un' operazione di veri- fica di un programma su disco o su nastro e' stato incontrato un errore.

CAPITOLO 9

GLASSARIO MINIMO

GLOSSARIO DEI TERMINI TECNICI E NON.

Introduciamo brevemente questo glossario, per anticiparvi che ciascuna delle definizioni date al suo interno e' principalmente formata di due parti: una prima parte tecnica che sintetizza il significato del termine di turno, ed una seconda parte che potremmo definire spiritosa e che non e' altro che il modo smaliziato di vedere le cose dell'informatica da parte di un addetto ai lavori.
Spesso vi sembrera' di non capire lo humor contenuto nella definizione: non preoccupatevi, verra' anche per voi il tempo di ridere allegramente di cose e situazioni che oggi (visto che state imparando a muovere i primi passi nel campo dell'informatica, e la cosa non e' obiettivamente cosi' semplice come molti vogliono far credere) vi causano notevoli mal di testa.
A proposito dell'addetto ai lavori che ha voluto farvi un po' sorridere, dobbiamo riportare anche la definizione che egli ha dato di se stesso... e scusateci se non e' in ordine alfabetico!

EDP MANAGER

Chi governa un grosso centro elaborazione dati. Il suo "personal" e' un computer che costa cifre a 9 zeri, pagate dalla sua societa' che paga profumatamente anche lui. Colui che ha scritto la seconda parte di ognuna di queste definizioni, rifiutandosi di credere alla prima...

BASIC

Linguaggio di programmazione ad alto livello (vedi High Level Language), inventato per scopi educativi a Dartmouth. Facile da usare e da imparare, e' ora disponibile sulla maggior parte dei sistemi a microprocessore.

BACK UP

Copia di archivi dati e/o di programmi conservata generalmente su un diverso supporto fisico per prevenire la perdita dell'originale (causa calamite, figli, mogli, furti di amici 'gelosi'...).

BIT

E' l'unita' elementare che si puo' riconoscere nella memoria di un computer: un bit e' un elemento che e' in grado di assumere lo stato di acceso o di spento (che stanno per si' e no), cioe' puo' assumere solamente due configurazioi diverse. Gruppi di bit variamente combinati costituiscono sia istruzioni di programma che dati da elaborare o elaborati.
I bit tendono a metter su famiglia, generando cosi' i bytes (vedi oltre).

BOOTSTRAP

"CALCIONE" letteralmente.
Sistema innato utilizzato per l'accensione del calcolatore. Generalmente azzera la memoria, predispone gli organi di I/O in modo adeguato e carica il sistema operativo da disco.
E' noto che se un calcolatore non lo si prende fin dall'inizio a calci, non va...

BUFFER

Dispositivo hardware in grado di ristabilire i livelli logici di pilotaggio dei segnali, onde poter pilotare un bus o un grande numero di ingressi.
E' definibile anche come area di parcheggio in cui i dati si fermano in attesa che il 'cervello' del calcolatore, momentaneamente impegnato altrove, decida dove mandarli.
E' una cosa cosi' complicata che per il momento non vi serve a niente sapere cos e'.

BUG

Errore. Debugging e' l'operazione di ricerca e correzione dell'errore.

Abbiamo tentato di sottoporre questo libro a un'operazione di debugging e speriamo che essa abbia avuto un discreto successo. In italiano BUG si chiama anche "baco", perche' un programma pieno di bachi e' come una mela bacata: "no buona"

BUS

Percorso di segnali aventi funzioni comuni. Ogni unita' a microprocessore di tipo standard ha tre tipi di bus: bus dati, bus indirizzi e bus di controllo.
Il biglietto si paga a chi vi vende il calcolatore...

BYTE

E' la piu' piccola unita' di memoria alla quale un programma e' conveniente che acceda. E' generalmente formato da otto bit. In un qualunque elaboratore ogni lettera alfabetica A, B, C etc., ogni cifra numerica 0,1,2 etc., ogni simbolo particolare -,*,$ etc. sono rappresentati da un byte, cioe' da una particolare combinazione di 8 bit presi nel loro insieme. Un byte puo' anche essere considerato l'unita' di misura della grandezza della memoria centrale di un elaboratore.
I calcolatori impazziscono quando perdono i bytes in memoria. Inutile sprecare tempo a cercarli: nessuno li ha mai trovati!

CARRIAGE RETURN

Tasto standard della macchina da scrivere che permette il ritorno a capo della testina.
Allontanare le dita dalla stampante ad ogni CARRIAGE RETURN e poi riavvicinarle e' uno dei piu' usati esercizi ginnici per programmatori.

CPU

Central Processing Unit.
Unita' centrale di elaborazione. Ovvero: IL CERVELLO.
La' dentro accadono cose strane per cui 2+2 fa davvero 4, e, se fa qualche volta 5, basta spegnere e riaccendere e il risultato e' nuovamente 4.

DATA BASE

Organizzazione razionale dei files dati, che consente

facile accesso, aggiornamento, lettura e selezione all'interno di un archivio dati.
Non per deludervi, ma dobbiamo confessarvi di non aver mai visto un data base che rispetti tali caratteristiche: a voi l'arduo compito.

DEBUGGING

Vedi BUG.
Ovvero e' il sistema per togliere i BUG dai programmi (un consiglio: se il programma non l'avete fatto voi, e' piu' semplice che ne compriate un altro)

DIGITARE

Scrivere tramite tastiera. Dire "digita A" significa: "premi il tasto corrispondente alla l.ttera A".
Modo elettronico moderno per complicare le cose semplici.

DISKETTE

Vedi FLOPPY DISK.

DIRECTORY

Tabella gestita dal sistema il cui contenuto permette di andare a files specifici.
Si chiama cosi' anche l'elenco degli archivi e dei programmi presenti su un Floppy. I nomi di tali archivi vengono dati da chi li scrive e sono solitamente tanto strani che, a distanza di tempo, non si riesce piu' a capire cosa c'era dentro, mentre quando li si inventa sembrano chiarissimi riguardo al loro contenuto.

DOS

Disk Operating System. Sistema operativo disco.

DRIVE

(Device). Parte fondamentalmente meccanica, con integrati elementi elettronici, richiesti per fornire i comandi base a un dispositivo come il floppy disk. Puo' essere composto di vari motori, sensori di posizione, luci, interruttori.
Gira sempre, e quando non gira piu' lui gira qualcos'altro...

DUMP

Operazione che permette di trasferire i contenuti di una memoria ad esempio su disco e/o su stampante.
I programmatori si riconoscono dagli immensi tabulati pieni di dump, presenti sui loro tavolini da notte, mobiletti dei bagni, tappeti etc.

FILE

Blocco logico di informazioni, rappresentate con un nome e considerate dall'utilizzatore come un unico elemento.
Un file puo' essere diviso fisicamente a seconda di come richiesto dal dispositivo di memoria in record, blocchi, o altre unita'.
Talune "divisioni" di file avvengono anche per caduta di corrente, taglio di floppy, errori di programma: in questi casi si parla di "file a pezzi".

FLOPPY DISK

Il termine significa esattamente "disco flessibile". E' un tipo di supporto magnetico realizzato per la memorizzazione di programmi e di dati. Consiste in un disco di plastica speciale, contenuto in una busta che lo preserva da sporcizia e rotture e che presenta tacche per la lettura. Il floppy disk e' un supporto diffusissimo per il favorevole rapporto prezzo/prestazioni e per l'alta affidabilita' e praticita' d'uso.
Serve bene in alternativa al "frisbee" o al disco per atleti poco muscolosi.

FLOW CHART

Rappresentazione simbolica della sequenza di un programma. I rettangoli indicano ordini o calcoli. I rombi rappresentano tests o decisioni (diramazioni). Il flow chart e' il passo intermedio tra le specifiche dell'algoritmo e la scrittura del programma. Esso facilita enormemente la comprensione e la correzione, permettendo di dividere il programma in moduli logici sequenziali.
Raramente si vede un FLOW CHART che rispecchi quello che POI fa il programma!...

HARD COPY.

Stampa su carta di quanto presente sul video.

HIGH LEVEL LANGUAGE

Linguaggio di programmazione molto simile alle parole di uso comune, dotato di istruzioni potenti. Per esempio FORTRAN - BASIC - APL - ALGOL - COBOL - PL1 sono linguaggi ad alto livello. Un HLL richiede un compilatore o un interprete.
Richiede anche che chi lo usa abbia voglia di piangere ogni notte (dubbio: non sara' mica stato inventato dai produttori di collirio, che vedono peraltro di buon occhio i programmatori nottambuli con occhi gonfi e pendenti?...)

INPUT - OUTPUT

Abbreviato I/O.
Collegamenti o dispositivi usati per ottenere informazioni dall'esterno o per dare tali informazioni al calcolatore come tastiere, stampanti, video, e chi piu' ne ha piu' ne metta...

INTERFACCIA

E' l'hardware/software richiesto per connettere un dispositivo o periferica ad un calcolatore. Esistono ora, per molte periferiche, interfacce realizzate tramite un solo chip.
Tutti i calcolatori spillano soldi ai loro ignari possessori perche' vogliono un sacco di interfacce, complici alcuni subdoli rivenditori...

K

Kappa e' l'iniziale di Kilioi, che in greco significa 1.000. K e' un modo piu' sintetico per contare il numero di bytes della memoria centrale di un elaboratore. K sta per K-Byte e corrisponde a 1.024 bytes. Mega-Byte e' l'unita' di misura superiore al K e sta ad indicare 1.048.576 bytes.
I kappa sono un po' come i soldi e le donne; piu' ce n'e' meglio e' sia per il calcolatore che per te (anche la rima: mah...).

KEY

Chiave. Nel gergo Key sta spesso ad indicare un tasto. Ad esempio: PRESS ANY KEY TO START e' da tradurre PREMI QUALSIASI TASTO PER PARTIRE.
(normalmente non succede nulla e se succede qualcosa non e' quella voluta. I soliti programmatori!)

KEYBOARD

Insieme di tasti o pulsanti usati per introdurre informazioni in un calcolatore. La Keyboard - Tastiera - e' un dispositivo di input.
Se si mette a sputare fuori i tasti e' perche' l'avete programmata come dispositivo di output...

KEYWORD

Parola chiave.
Serve a far si' che rimanga segreto il modo di utilizzo di una funzione o di un programma. Spesso la si dimentica e prima di poter usare la funzione protetta si piange.

LINE FEED

Comando di avanzamento verso l'alto della carta, pari ad una riga (o del cursore verso il basso in uno schermo).
Ci sono stampanti con l'auto-line feed e stampanti senza lo stesso.
Se si da' il line feed ad una stampante con auto line feed si possono avere fino a 10 piani di ... morbidezza (dipende sempre dalla carta).

LOOP

Insieme di istruzioni che possono essere eseguite piu' di una volta, in cicli omogenei successivi.
Normalmente il primo loop non si scorda mai, anche perche' se ci si pensa su troppo si rischia di restarci per sempre.
Esempio di loop:

```
10 DOVE VAI?
20 AL CINEMA
30 A VEDERE COSA?
40 QUO VADIS
50 COSA VUOL DIRE?
60 GOTO 10
```

MONITOR

Programma o insieme di programmi che permettono di interpretare i comandi fondamentali richiesti per l'uso di un sistema.
Viene chiamato cosi' anche il "televisore" collegato al vostro calcolatore.

ON-LINE

Contrario di OFF-LINE. Si dice di un dispositivo collegato ad un calcolatore e pronto a ricevere dal calcolatore stesso dati e informazioni o a mandargliene.

OPERATING SYSTEM

E' il software richiesto per gestire le risorse hardware e logiche di un sistema, incluso la gestione e l'ordinamento di un file (vedi FILE).

PACKAGE

Programma o insieme di programmi sviluppato per una specifica applicazione ed utilizzato da vari utenti.
Viene prodotto da altri per farvi impazzire per capire come loro non hanno capito cosa voi volevate che capissero.

PERIFERICHE

Sono le unita' costituenti con l'Unita' Centrale (U.C.), ma staccate da questa, l'elaboratore. La stampante, ad esempio, e' una periferica. Le periferiche possono ricevere dati dall'U.C. o inviarli a loro volta.
I calcolatori sono felicissimi di avere tante periferiche, visto che sono di genere femminile, ma cio' provoca la gelosia delle mogli che vorrebbero spendere altrimenti i soldi che voi destinate all'espansione del vostro sistema.

PLOTTER

Dispositivo meccanico in grado di disegnare linee sotto il controllo di un elaboratore.

PROGRAMMA

E' una serie di istruzioni date al computer che gli fanno eseguire un determinato lavoro. Ad esempio per far emettere delle fatture ad un elaboratore, occorre che gli sia stato immesso prima un programma "di lavoro" che lo guidi nell'emissione delle fatture: da solo il computer non sa fare niente!... Pensateci.
E a forza di pensarci, a forza di dump, a forza di bugs, a forza di collirio, a forza di unghie rotte sui tasti, a forza di capelli strappati, a forza di pugni sul monitor che dovrebbe essere il classico "ambasciator che non porta pena" visto che e' l'unico che non ne puo' proprio niente, riuscirete a fargli fare 2+2=4

ROUTINE

Programma o parte di un package o di un programma con funzioni specifiche (ad esempio e' tipica la Routine di controllo Input).
Altrettanto tipica e' la routine "inlooppata" o "bacata", che tanta gioia provoca in voi e al vostro programma.

SCROLLING

Movimento verso l'alto o il basso pari ad una linea del contenuto dello schermo.

UTILITIES

Programmi o routines di frequente utilizzo. Ad esempio: copia dischetti, formattamento, lista files etc.

WORD PROCESSOR

Indica programmi per il trattamento dei testi. Tali programmi sono descritti in modo (speriamo) esauriente in un paragrafo di questo libro.

CAPITOLO 10

PROGRAMMA DIMOSTRATIVO

CLASSIFICA IN TEMPO REALE

Un po' di calcio sara' necessario, assieme ad una sana cura a base di ricostituenti, per riabilitarvi dopo la lettura di questo libro. Il programma che vi presentiamo, visto che il calcio e' uno sport molto amato e diffuso, vuole essere un pretesto per farci perdonare da tutti coloro che avranno trascorso il loro tempo maledicendo il libro e chi l' ha scritto.
La sua funzione e' quella di mantenere la classifica aggiornata durante i 90-150 minuti di sofferenza (questa volta non elettronica ma sportiva) domenicale, in cui i punteggi variano in continuazione senza dare un attimo di tregua.
Il vostro COMMODORE 64 ,acceso in precedenza e caricato con il numero della giornata, con gli incontri ed i punteggi delle squadre, vi aiutera' a seguire lo svolgersi della classifica minuto per minuto, bastera' aggiornarlo ogni volta che avrete notizia di un risultato (cosi' oltre ai soliti innumerevoli canali video che vengono tenuti in movimento per poter scrutare l'aria in attesa di goals, si aggiungera' anche il num. 36 per il calcolatore).
Tenete presente che la classifica illustrata sul video parte assegnando un punto di pareggio ad entrambe le squadre, fino a quando una segnatura non modifichi le sorti dell' incontro.
Se vi accadesse di perdere il controllo video, battete VID al posto del nome di una squadra e potrete ripristinare l' immagine senza perdere i punteggi.
END vi consentira' di uscire dal programma (e se volete divertirvi a programmare potrete aggiungere una piccola SUB che salvi i punteggi per la settimana successiva).
Le istruzioni contenute nel programma sono cose che avete gia' avuto modo di vedere in precedenza, dovete quindi essere in grado di capire il programma cosi' come e' scritto senza bisogno di ulteriori commenti; se cio' non vi sara' possibile, non avrete che da seguire l' ultimo penosissimo consiglio:

RESTORE: GOTO CAP1: READ: READ: READ: READ.........

```
5 REM   G% NUM GIORNATA
10 REM  N  NUM DI SQUADRE
15 REM  M  NUM DI INCONTRI
20 REM IN$(M,1)  SQUADRA IN CASA
21 REM IN$(M,2)  SQUADRA OSPITE
25 REM CL$(N)  CLASS. SQUADRE BASE
26 REM CO$(N)  CLASS. SQUADRE RIELAB
30 REM CL%(N)  PUNTEGGIO CLASS. BASE
31 REM CO%(N)  PUNTEGGIO CLASS. RIELAB
35 REM GF%(N)  GOALS FATTI BASE
36 REM FO%(N)  GOALS FATTI AGGIORNATO
40 REM GS%(N)  GOALS SUBITI BASE
41 REM SO%(N)  GOALS SUBITI AGGIORNATO
50 REM           *
60 REM
79 REM - DIMENSIONAMENTO
80 REM
90 POKE 53280,0:POKE 53281,0
100 N=16:M=8:G%=12
120 DIM IN$(M,2),IN%(M,2),CL$(N),CO$(N),CL%(N),CO%(N)
130 DIM GF%(N),FO%(N),GS%(N),SO%(N),DO%(N)
150 REM - CARICAMENTO DATI INIZIALI
156 F5$="TTTTTTTTTTTTTTTTT":REM 17 SHIFT CRSV
170 GOSUB 1100
183 TI$="000000"
185 PRINT"□";
190 L$="++++++++++++++++++++++++++++++++++++++++":REM 40
193 B1$="                     ":REM 21
195 B2$="                ":REM 16
196 REM STAMPA TESTATA
197 PRINTL$;
200 PRINT"■ CAMPIONATO ITALIANO DI CALCIO SERIE  A□"
210 PRINT" ";"■";G%;"■.MA GIORNATA□";TAB(24)"ORA M/S -";TAB(33)TI$;"-";
220 PRINTL$;:PRINT"■ANALISI RISULTATI□     +   ■CLASSIFICA□"
230 PRINT"■  E CLASSIFICA□         +      ■PUNTI DIFF  □";
240 PRINT"■IN TEMPO REALE   □     +            ■RETI  □";
245 PRINT LEFT$(L$,22)
248 PRINT"T"
250 FORI=1TO16:PRINTTAB(22)"+" :NEXT
260 PRINT"TTTTT" LEFT$(L$,22);
280 PRINT"TTTTTTTTTT":FORI=1TOM
290 PRINT"■" IN$(I,1);TAB(9-LEN(STR$(IN%(I,1))))IN%(I,1);
300 PRINTTAB(12)IN$(I,2);TAB(21-LEN(STR$(IN%(I,2))))IN%(I,2);"□"
310 NEXT
320 REM - ROUTINE PRINCIPALE
330 GOSUB 360
340 GOSUB 600
350 GOTO 330
360 REM - RICALCOLO CLASSIFICA
370 FORI=1TON:CO$(I)=CL$(I):NEXT
380 FORI=1TOM:J=I*2-1:L=I*2
390 FO%(J)=GF%(J)+IN%(I,1):SO%(J)=GS%(J)+IN%(I,2):DO%(J)=FO%(J)-SO%(J)
400 FO%(L)=GF%(L)+IN%(I,2):SO%(L)=GS%(L)+IN%(I,1):DO%(L)=FO%(L)-SO%(L)
410 IF IN%(I,1)=IN%(I,2)THEN CO%(J)=CL%(J)+1:CO%(L)=CL%(L)+1:GOTO 440
420 IF IN%(I,1)> IN%(I,2) THEN CO%(J)=CL%(J)+2:CO%(L)=CL%(L):GOTO440
430 IF IN%(I,1)<IN%(I,2) THEN CO%(J)=CL%(J):CO%(L)=CL%(L)+2
440 NEXTI
```

```
450 REM - ROUTINE ORDINAMENTO CLASSIFICA
460 FORI=1TON-1:FORL=I+1TON
470 IF CO%(I)>=CO%(L)THEN 510
480 CO%=CO%(I):DO%=DO%(I):CO$=CO$(I)
490 CO%(I)=CO%(L):DO%(I)=DO%(L):CO$(I)=CO$(L)
500 CO%(L)=CO%:DO%(L)=DO%:CO$(L)=CO$
510 NEXT L,I
520 REM - SCRITTURA DELLA CLASSIFICA
525 PRINTLEFT$(F5$,11)
530 FORI=1TON:S$="+":IFDO%(I)<0THENS$="-"
540 IFDO%(I)=0THENS$=" "
550 DO%(I)=ABS(DO%(I))
560 PRINTTAB(23)B2$
570 PRINT"▮";"▯"TAB(23)CO$(I);"◣";TAB(33-LEN(STR$(CO%(I))))CO%(I);
580 PRINT"▯";TAB(34);S$;"◣";TAB(36-LEN(STR$(DO%(I))))DO%(I);"▯"
590 NEXT:RETURN
600 REM INPUT RISULTATI PARTITE
610 PRINT"▯▯";"▮IMMETTI CODICE▯"
620 PRINT"▮SQUADRA CHE SEGNA▯"
630 PRINT"   ▮+++▯              "
645 PRINT"▯";:Q$="":SS$=""
646 GETQ$:IFQ$=""THENGOSUB752:GOTO646
647 SS$=SS$+Q$
648 IF LEN(SS$)=3THENPRINT"▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮";SS$;:GOTO650
649 GOTO646
650 IF SS$="VID"THEN185
651 IF SS$="FIN"THEN END
660 REM CERCA LA PARTITA DEL GOAL
670 FOR I=1TO8
680 IF LEFT$(SS$,3)=LEFT$(IN$(I,1),3)THEN IN%(I,1)=IN%(I,1)+1:GOTO740
690 IF LEFT$(SS$,3)=LEFT$(IN$(I,2),3)THEN IN%(I,2)=IN%(I,2)+1:GOTO740
720 NEXT:PRINT"▯▯▯";"▮CODICE ERRATO▯":PRINT"▯▯": FORK=1TO999:NEXT
721 PRINT"                      ▮▮":GOTO610
739 REM - SCRITTURA ULTIMO RISULTATO:GOTO748
740 PRINT:PRINTLEFT$(F5$,16-I);
748 PRINT"▮";IN$(I,1);TAB(9-LEN(STR$(IN%(I,1))))IN%(I,1);
750 PRINTTAB(12)IN$(I,2);TAB(21-LEN(STR$(IN%(I,2))))IN%(I,2)
751 PRINT"▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮";:RETURN
752 PRINT"▮"
753 REM - OROLOGIO
755 PRINT"▮";TAB(33)TI$;"▯"
780 RETURN
1100 REM  INPUT DATI DA TASTIERA
1110 PRINT"▯▮▮▮PROGRAMMA PER CLASSIFICA"
1120 PRINT"▮▮IN TEMPO REALE "
1130 PRINT"▮▮BY BRANCO PAVIN"
1140 PRINT"▮▮▮▮DATI DA INSERIRE:"
1141 F1$="▮SQUADRA CHE GIOCA IN CASA"
1142 F3$="▮PUNTI IN CLASS.,GOALS FATTI,GOALS SUBITI"
1143 F2$="▮SQUADRA OSPITE"
1150 PRINTF1$
1160 PRINTF3$
1170 PRINTF2$
1180 PRINTF3$
```

```
1190 INPUT"NUM GIORNATA";G%
1200 FORI=1TOM:J=I*2-1:L=2*I
1205 PRINT"{CLR}{DOWN}{DOWN}"
1210 PRINT"PARTITA NUMERO";I;"{DOWN}{DOWN}"
1220 PRINTF1$;:INPUTIN$(I,1)
1230 PRINTF3$:INPUTCL%(J),GF%(J),GS%(J)
1240 PRINTF2$;:INPUTIN$(I,2)
1250 PRINTF3$:INPUTCL%(L),GF%(L),GS%(L)
1251 GOSUB5010
1252 CL$(J)=IN$(I,1):CL$(L)=IN$(I,2)
1260 NEXT
1270 FOR I=1TOM:J=I*2-1:L=I*2
1280 PRINTIN$(I,1);TAB(7)CL%(J);GF%(J);GS%(J);
1290 PRINTTAB(20)IN$(I,2);TAB(27)CL%(L);GF%(L);GS%(L)
1300 NEXTI
1305 REM CORREZIONI DATI INIZIALI
1310 INPUT"{DOWN}{DOWN}{DOWN}VUOI CORREGGERE I DATI";Q$
1320 IFLEFT$(Q$,1)="N"THENRETURN
1330 IFLEFT$(Q$,1)<>"S"THEN1310
1340 INPUT"{DOWN}{DOWN}NUMERO DI PARTITA";I:J=I*2-1:L=I*2
1350 PRINTF1$;:INPUTIN$(I,1)
1360 PRINTF3$:INPUTCL%(J),GF%(J),GS%(J)
1370 PRINTF2$;:INPUTIN$(I,2)
1380 PRINTF3$:INPUTCL%(L),GF%(L),GS%(L)
1391 GOSUB5010
1392 CL$(J)=IN$(I,1):CL$(L)=IN$(I,2)
1400 GOTO1270
2280 Y.
5000 REM RIDUCE I NOMI A 7 CARATTERI
5010 IN$(I,1)=LEFT$(IN$(I,1),7)
5020 IN$(I,2)=LEFT$(IN$(I,2),7)
5030 RETURN
```

Nato dalla felice collaborazione di un ingegnere che usa COMMODORE da anni e di un operatore del settore di estrazione tecnica, questo libro vi guiderà alla scoperta delle capacità del vostro COMMODORE 64, dall'apertura della scatola alla programmazione della grafica e del suono.
La chiarezza espositiva e la gradualità seguita nel trattare i diversi argomenti, fanno di questo testo uno strumento fondamentale per la comprensione del COMMODORE 64 e della sua programmazione.
Accenni di carattere generale ed un divertente glossario fanno sì che anche il più inesperto degli utilizzatori possa entrare nel mondo affascinante dei computers e imparare i rudimenti della programmazione in linguaggio BASIC.
Anche il tono a volte scherzoso contribuisce a sdrammatizzare gli argomenti più ostici.
Alcuni utili consigli su come trattare il vostro COMMODORE 64 e l'accenno a programmi di uso generico ne fanno un testo davvero completo.
Tra i molteplici programmi contenuti nel libro (tutti provati dagli autori) che voi potrete trascrivere ed usare, sperimtenando così la programmazione in BASIC, anche un programma che consente di tenere la classifica delle squadre di calcio in tempo reale.

L. 22.000 Cod. 347D ISBN 88-7056-033-3

100
VOI E IL VOSTRO COMMODORE 64
Fulvio Francesconi
Fernando Paterlini
GRUPPO
EDITORIALE
JACKSON